BOOKS ... II, III, IV

Minivac 601

What is a Digital Computer?

How Computers Make Logical Decisions

How Computers do Arithmetic

MINIVAC 601

BOOKS II-III-IV

SCIENTIFIC DEVELOPMENT CORP.
WATERTOWN, MASS.

The Minivac Manual was prepared and edited by the staff of

Scientific Development Corporation

First Printing—August, 1961

CONTENTS

BOOK II: WHAT IS A DIGITAL COMPUTER?

BOOK III: HOW COMPUTERS MAKE LOGICAL DECISIONS

BOOK IV: HOW COMPUTERS DO ARITHMETIC

BOOK II

What is a Digital Computer?

PREFACE

This is the second in a series of books using MINIVAC 601 to explore the world of "electronic brains". In writing this book, the authors have assumed that the reader is familiar with the information contained in the first book of this series and understands the operation of the components of MINIVAC 601.

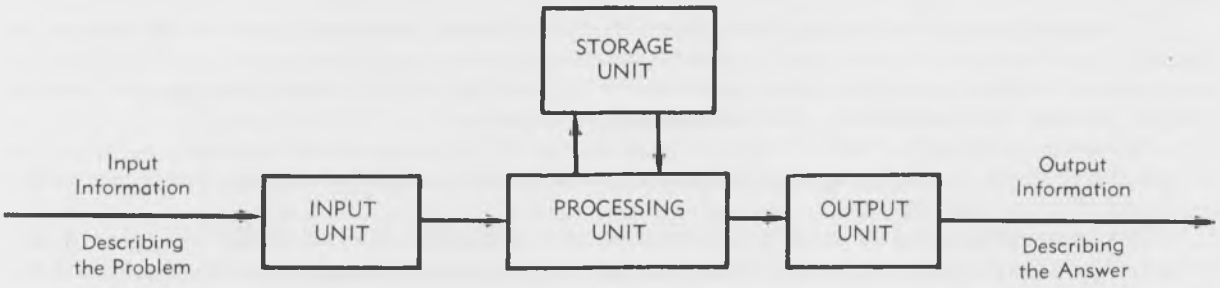
The basic question which this book was written to answer is "What is a digital computer?" In order to answer this question it is necessary to examine the functions and forms of modern high-speed digital computer systems. This book describes the major characteristics of modern computer systems and compares the functions performed by the components of the MINIVAC 601 with those performed by similar parts in a large scale digital computer.

I. INTRODUCTION

In order to function as a digital computer a machine or combination of machines must be able to handle information or "data" in an orderly manner. It must be able to receive information as "input" from the outside world. Once received, this information must be "processed" by the computer, and the result of the processing must be "remembered" or "stored" for future use. After an answer has been obtained, it must be communicated back to the outside world as "output." Thus a general purpose digital computer is made up of four basic units:

The input unit
The processing unit
The storage unit
The output unit.

The basic units are connected together like this:



FLOW CHART OF COMPUTER OPERATION

Input Unit

Information about a particular problem must be given to the computer before any operation can be performed. Input information may be of two kinds.

(1) Data: The numbers or coded information to be used in calculation are called ***input data.*** These numbers may represent physical measurements, mathematical relationships, or conditions of a "logical" decision-making problem.

(2) Instructions: The computer must be instructed to perform specific operations in a definite sequence. Input information which directs the computer to perform certain operations and to handle the data in a specified way is called the ***instructions.***

Input information is supplied to MINIVAC 601 through the binary input pushbuttons and the decimal input-output rotary switch. Instructions are communicated to MINIVAC 601 by wiring on the computer console a "program" which instructs the computer to perform certain operations. The binary input pushbuttons are designed to communicate zeros and ones to the computer and the decimal input-output rotary switch is designed to communicate decimal numbers to the computer.

A large scale digital computer, such as the IBM 7090 illustrated later in this section, may receive input information directly through pushbuttons similar to those used on MINIVAC 601 or it may receive input information through punched cards or magnetic tape. Some computers receive input information through punched paper tape; others receive direct input information through a typewriter-like unit called a "flexowriter".

Later in this book, each type of input unit found on a large electronic data processing machine will be discussed and compared with the input devices of MINIVAC 601.

Processing Unit

The processing unit of a digital computer performs four major functions:

(1) Control: the processing unit controls the operations of the computer system and interconnects the input, output and storage units. All calculations, operations, and information transfer are accomplished under "control" of the processing unit.

(2) Decision-Making: the processing unit is able to perform comparisons which are the basis of all computer decision-making. By comparing two numbers or symbols with each other and determining whether or not they are equal, the computer decides upon a course of action.

(3) Arithmetic: in the processing unit of a digital computer all normal arithmetic functions are performed. These operations are actually done in a part of a processing unit known as the "arithmetic unit" which is designed to perform addition, subtraction, multiplication and division.

(4) Logic: the processing unit of most high-speed digital computers is equipped to perform various "logical" operations through which conclusions of a non-arithmetic type may be reached. Just as arithmetic operations provide the steps by which the solution to a mathematical problem is reached, the logical operations provide the steps in a "reasoning" process.

The third book in this series, ***How Computers Make Logical Decisions,*** describes the nature of logical operations and decision-making functions through demonstration on MINIVAC 601. The nature of arithmetic operations performed in the processing unit of large computers is demonstrated in detail in the Book IV: ***How Computers Do Arithmetic.***

The processing unit of MINIVAC 601 is made up of six relays and the rotary switch. The relays and the rotary switch are used to provide control, make decisions, and perform basic arithmetic and logical operations.

The processing units of most large scale digital computers use advanced electronic components to perform the functions demonstrated by the relays and rotary switch on the MINIVAC 601. The circuits of the processing unit in these machines use germanium or silicon transistors and diodes which are designed to perform millions of operations in one second.

In this book the important features and operating characteristics of the processing unit of a modern electronic data processing machine are described. The processing unit of the MINIVAC 601 is examined as a basic illustration of the functions of the processing unit of a modern digital computer.

The Storage Unit

In order to function efficiently, a digital computer must be able to "store" or "remember" data for use in processing and computation. Input information is "read" into storage under control of the processing unit and called from storage as it is required for use by the processing unit.

In the course of a normal high-speed digital computer program, the processing unit of the computer follows a series of ***instructions*** which are stored in the storage unit using ***data*** which is also stored in the storage unit. Information obtained during calculations performed is stored temporarily in the storage unit for use at a later time.

A computer uses a storage unit in much the same way that we use a piece of paper when solving a long division problem. The partial answers to the long division problem are temporarily ***stored*** (written) on the paper until the complete answer is obtained. Similarly, the partial answers to the computer's problem are held in storage until the complete answer has been obtained. The answer which is to be communicated by the computer to the outside world as "output" may also be stored in the storage unit until it is sent by the processing unit from storage to the output device.

Several different methods of storage are used in modern computer systems to enable the machine to "remember" instructions, data, partial and final results of calculations, and output information. In smaller machines and the MINIVAC 601, the processing unit is also used for storage. The relays of MINIVAC 601 supply the major source of operating storage. These "memory units" of the MINIVAC 601, although smaller than those found in commercial computers, demonstrate the way in which information is stored in a digital computer.

The decimal input-output unit also serves as a storage device during the operation of some programs. The rotary switch remains in a particular position until it is moved and thus enables MINIVAC 601 to "remember" a decimal number.

Large scale digital computers may use relays for storage and, in this case, be identical to the MINIVAC 601. Among the most popular storage devices in use today are the coincident-current magnetic core, magnetic drum, and magnetic disc storage units. These storage systems are all used for "operating storage"—temporary data storage ***within*** the computer while a program is being "run". Other storage media are used to store data and instructions for longer periods of time and to save data and instructions ***outside*** the computer. These "permanent" storage media include magnetic tape, paper tape and punched cards. Each of these storage methods will be discussed later in this book and the relays and rotary switch of the MINIVAC 601 will be used to illustrate the theory and operation of each type of modern storage device.

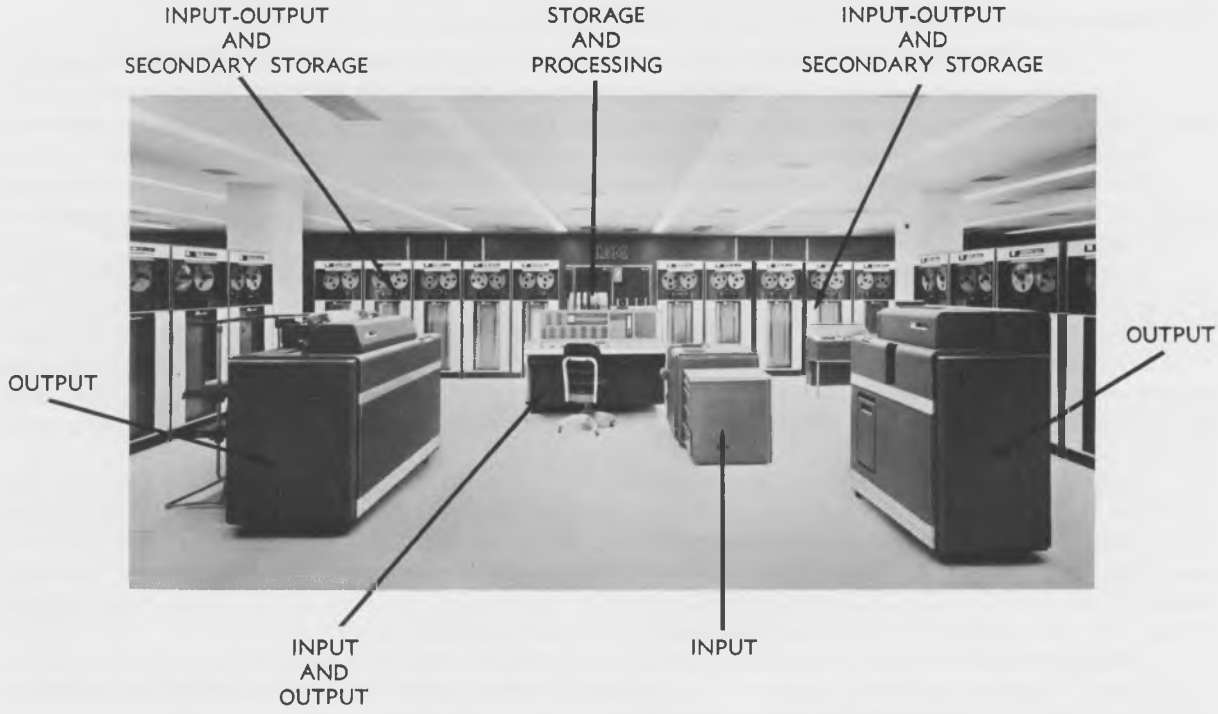
Output Unit

Output information generated by a digital computer is communicated to the outside world to present the answer to a problem or to describe the operations of the computer. Output media used in modern digital computer systems include magnetic and paper tape, and punched cards. Some auxiliary output units convert numerical output obtained from the computer to charts, graphs, photographic displays, and numbers or letters on a printed page. The great diversity of output devices which have been developed for use with the modern high-speed digital computer makes it possible for information to be presented by the computer in almost any form in which it is desired.

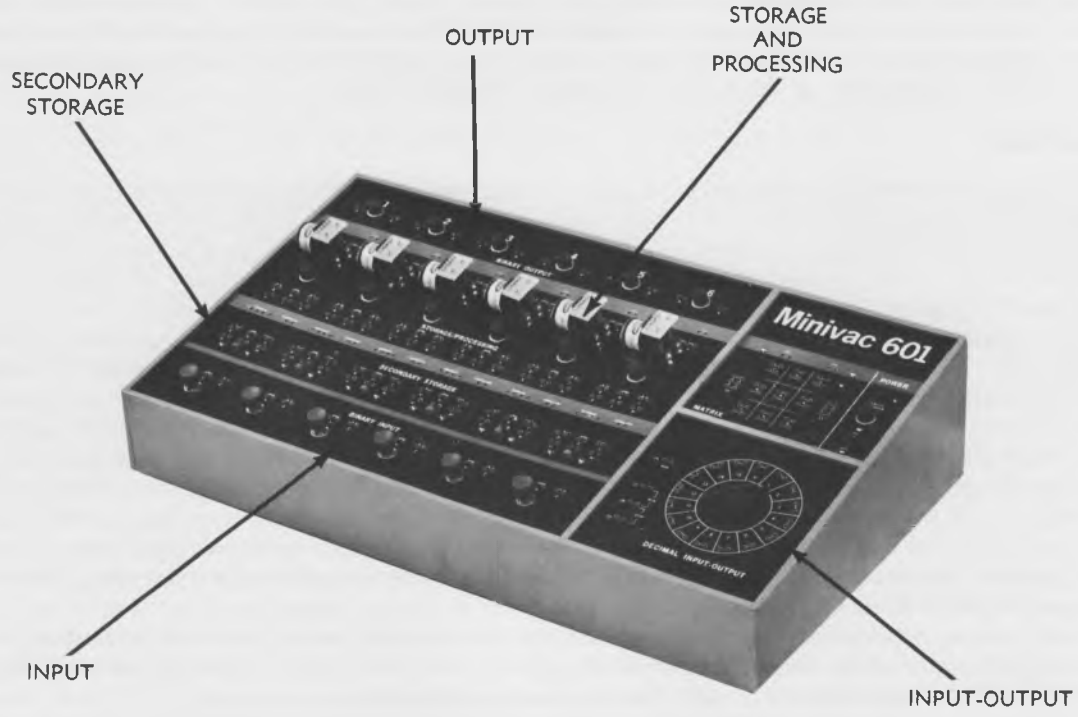
The output devices of the MINIVAC 601 are the binary output lights and the decimal input-output rotary switch. When operated with the motor, the rotary switch becomes an output device, since the motor may be controlled by the processing unit of the MINIVAC 601 and caused to stop with the pointer indicating the number which is to be communicated as output information. When the binary output lights are used, output information is communicated in a special binary code discussed in detail in Book IV. Modern electronic data processing machines also have binary output lights for direct communication of information from the computer using the same binary code employed by MINIVAC 601.

Large scale digital computers employ output units using media previously discussed in conjunction with input units. Media employed for both input and output purposes include magnetic tape, paper tape, punched cards and data transmission links.

Various output units are discussed in detail later in this book and the output devices of the MINIVAC 601 are used to illustrate the nature of each type of output unit.



IBM 7090 DATA PROCESSING SYSTEM



MINIVAC 601

2. BASIC COMPUTER FUNCTIONS AND MINIVAC 601

The Basic Input Function

Human Input Functions

The input function can be considered in terms of the activities of a human being who, having no previous training in mathematics, is asked to perform a division problem. Consider this person sitting at a table with pencil and paper in hand. ***Input information*** is all information which must be communicated to the person before he can perform the division problem.

First, he must be given two numbers—the ***data***. He must be given the number which is to be divided (the dividend) and the number by which the dividend is to be divided (the divisor). In addition to this ***data input*** the person, since he knows nothing about arithmetic, must be told ***how to proceed*** with the numbers which he has received as data in order to obtain the answer. In short, the person must be given ***instructions*** on how to proceed to solve the problem. The instructions must be very detailed. They must tell him how to handle each number and how to proceed through each step of the process of division.

A basic computer which has not been equipped with any operating circuits is in much the same position as the man who has never heard of arithmetic. Under such circumstances the computer, like the man, must be given very detailed information about how to proceed through the problem. Fortunately, a digital computer can be equipped with circuits which give it a sufficient "knowledge" of the rules of arithmetic so that given the divisor and the dividend, it can be told simply "divide" and it will proceed to obtain an answer.

Forms of Input

As has already been noted, input information may be of two kinds: instructions and data. Both kinds of information may be communicated to the computer in different ways. Most large computers are equipped to receive input information from several different media. For example, the IBM 7090 computer system is able to receive information from punched cards and from magnetic tape.

Information presented in each of the media may be communicated in several different codes. Data may be presented as decimal information using the characters 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Information in this form is communicated to MINIVAC 601 with the rotary switch.

Both data and instructions may be communicated to the computer using a "binary" code based on the two-valued (zero and one) code referred to in Book I. The binary number system is discussed in detail in Book IV. For the purposes of discussion in this book we will only need remember that binary codes involve only two characters, zero and one, while decimal codes use the ten characters noted above.

MINIVAC 601 Input Units

Input information is communicated to MINIVAC 601 through the two input devices located on the console.

Binary input is communicated using the six binary input pushbuttons according to the convention:

Button up = zero (0)
Button down = one (1)

For example, the binary number "101" (5) is communicated to MINIVAC by pushing down pushbuttons 4 and 6 while leaving pushbuttons 1, 2, 3, and 5 up. The largest binary number which can be communicated to MINIVAC 601 is 111111 (63).

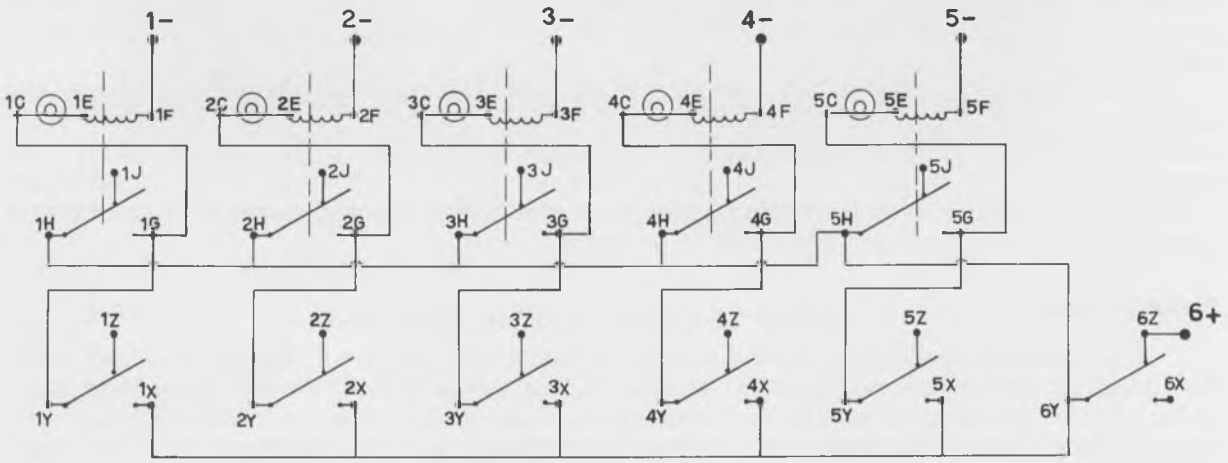
Decimal information is communicated using the decimal input dial and the pointer of the rotary switch. To give MINIVAC a decimal number as input, the pointer is turned so that it points at the desired digit. The decimal input capabilities of the MINIVAC 601 are limited to the numbers from zero through fifteen.

The following two experiments illustrate communication to MINIVAC in Binary and Decimal. Each experiment uses the relays to store (remember) the information communicated to the computer.

EXPERIMENT 1: BINARY INPUT

This experiment uses MINIVAC 601 to demonstrate how a digital computer receives binary input data and instructions. The binary number "one" is given to the computer by pushing a pushbutton ***down***. The binary number "zero" is given to the computer by leaving a pushbutton ***up***. The ***instruction*** to forget all previous data (to "clear" the memory) is given to the computer by pushing pushbutton 6.

The program and circuit drawing for this experiment are:



1X/2X	6Z/6+	2H/1H	2G/2C	4Y/4G	5F/5—
2X/3X	6Y/5H	1Y/1G	2F/2—	4G/4C	
3X/4X	5H/4H	1G/1C	3Y/3G	4F/4—	
4X/5X	4H/3H	1F/1—	3G/3C	5Y/5G	
5X/6Y	3H/2H	2Y/2G	3F/3—	5G/5C	

1. Turn power ON. Push pushbutton 1 and release. This transmits a "one" to section 1 of MINIVAC. The "one" is ***remembered*** by relay 1. Relay light 1 comes ON to indicate that a "one" is being remembered (stored). Data has been ***communicated*** from the operator to MINIVAC 601 by pushing BINARY INPUT pushbutton 1.
2. Do NOT push pushbutton 2. This leaves a "zero" in section 2 of MINIVAC 601. The "zero" continues to be remembered by relay 2. Relay light 2 remains OFF to indicate that a "zero" is being remembered (stored).
3. Do NOT push pushbutton 3. This leaves a "zero" in section 3. Relay light 3 remains OFF to indicate that a "zero" is being remembered.
4. Push pushbutton 4 to transmit a "one" to section 4. Relay light 4 comes ON to indicate that a "one" is being remembered by section 4.
5. Push pushbutton 5 to transmit a "one" to section 5. Relay light 5 comes ON to indicate that a "one" is being remembered by section 5.

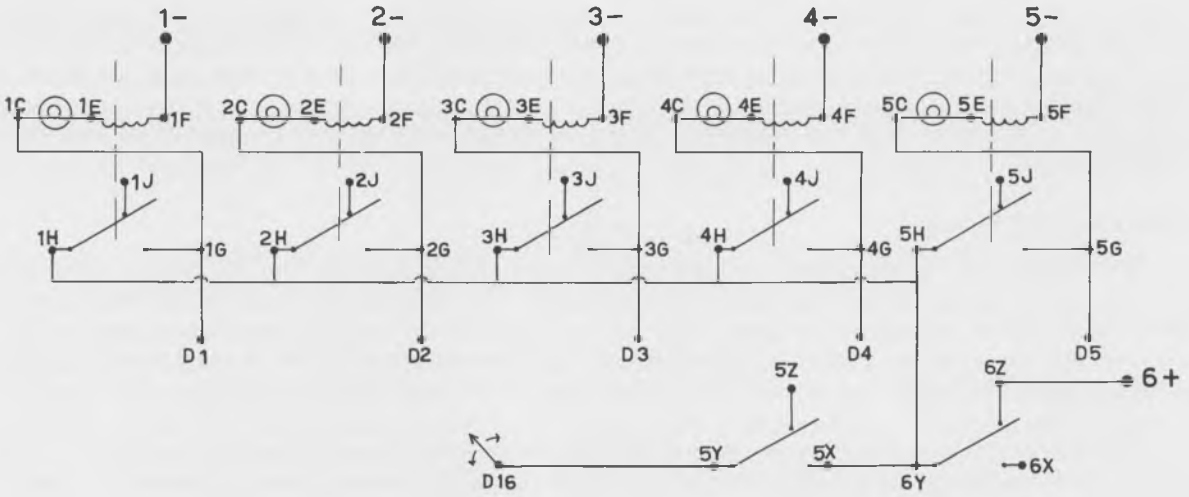
The binary number 10011 (19) has been communicated to MINIVAC 601 by using the BINARY INPUT pushbuttons. This input data is now being remembered (stored) in the first five sections of the computer.

6. Push pushbutton 6 and release. This action ***instructs*** the computer to forget all previous data. All relay lights go OFF, and the previous number is forgotten ("cleared" from the memory).

7. The computer is now ready to receive another binary number from the operator. Make up another number yourself and communicate it to the computer by using the BINARY INPUT pushbuttons.

EXPERIMENT 2: DECIMAL INPUT

This experiment demonstrates how MINIVAC 601 may receive decimal input data and instructions. A decimal number is communicated to the computer by turning the DECIMAL INPUT-OUTPUT knob to the desired number and pushing pushbutton 5 to instruct the computer to ***remember*** the selected number. Another instruction—to ***forget*** all previous data—is given to the computer by pushing pushbutton 6.
The program and circuit diagram for this experiment are:



1H/2H	6Z/6+	1F/1—	3G/3C	D5/5G
2H/3H	6Y/5X	D2/2G	3F/3—	5G/5C
3H/4H	5Y/D16	2G/2C	D4/4G	5F/5—
4H/5H	D1/1G	2F/2—	4G/4C	
5H/6Y	1G/1C	D3/3G	4F/4—	

1. Turn power ON. Turn the DECIMAL INPUT-OUTPUT knob to number 4 and push pushbutton 5. This transmits the number "four" to the computer. Relay light 4 comes ON to indicate that a "four" is being stored. Data has been communicated from the operator to MINIVAC 601 by turning the DECIMAL INPUT-OUTPUT knob to the desired number and by instructing the computer to ***remember*** the selected number by pushing pushbutton 5.

When **decimal** data is remembered by the computer, the relay lights have a different meaning than when ***binary*** data is being remembered. With binary data, the light ON represents a data "one" and the light OFF represents a data "zero". With decimal data, the number being remembered corresponds with the section (1-6) which has a relay light ON.

2. Push pushbutton 6 and release. This action ***instructs*** the computer to forget all previous data, and relay light 4 goes OFF. The computer is now ready to receive another decimal number (1-5) from the operator. Select another number yourself (between 1 and 5) and communicate it to the computer using the DECIMAL INPUT-OUTPUT.

The Basic Storage Function

Three types of storage are used in most high-speed data processing machines. These are:

Internal storage
Secondary storage
External storage.

Internal storage, which is for our purposes the most important, is storage available in a storage unit connected directly to the processing unit of the computer in such a way that the processing unit has "immediate access" to the information. Secondary storage refers to storage units in which the information is available to the processing unit but in which, due to the nature of the storage unit, it is available only after some delay. The difference between internal and secondary storage in thus the difference between immediate and delayed access to information.

External storage refers to storage in a media ***outside*** the computer system. External storage is thus accomplished by communicating the information to be stored out of the computer as "output" and then saving this information in "external storage" on the output media.

Human Storage Functions

The storage function can be clearly seen in the analogy of the human being with pencil and paper. In this case, the ***paper*** is the ***external storage system,*** and the pencil is the ***output device*** through which information is communicated from the human being to the external storage media. Internal and secondary storage are difficult to distinguish in the human case since we as human beings have only one storage system available for our use—the human brain. Information retained in the brain while the problem is being worked is probably best thought of as held in internal storage.

MINIVAC 601 Storage

MINIVAC 601 is equipped with internal storage in the form of the six relays which also serve as a part of the processing unit of the computer. The processing unit of MINIVAC does not have direct access to external storage. The human operator can supply external storage by writing information down on a sheet of paper when it is communicated to him through the output devices of MINIVAC and later return this information to the computer through the input devices as it is required.

Before examining the storage units of MINIVAC 601 and larger digital computers in more detail, it is necessary to consider the general form in which information is stored in a digital computer.

The Binary Nature of Storage

Just as information is communicated to the computer and processed by it in a binary form, it must be stored in ***binary*** form.

The reason for this use of the simple two-character (0 and 1) rather than the more complicated ten-character (0-9) decimal system in the computer can be realized by examining the decimal input device. Using the rotary switch mechanism involves many different physical positions which must be communicated by the switching system to the computer. The mechanism used to do this involves physical motion and several moving parts. In addition, the rotary switch system requires more than a second to go from zero to nine.

In contrast to the decimal system, the binary input button involves only one moving part and, since it has only two physical positions, is simpler in construction and can be moved from one position to another in a fraction of a second.

Thus, for the reasons of simplicity, efficiency and speed aptly demonstrated by the comparison between the decimal input switch and the binary input button, large computers handle all processing and storage using a binary system. Thus the basic element of storage in a computer is a binary "bit" which may have the value of 0 or 1.

The Structure of Storage

In any storage system, the ***bits*** are arranged into ***words*** consisting of a specified number of bits. Storage within the computer is handled in words. Each storage location—called a "register" contains one ***word*** of data and has a location number associated with it so that it is possible to identify a register and store information in or call information from that register.

The storage system of a large computer might be thought of as a large number of boxes with

each box identified by a number indicating its location in the series of boxes, with each box just the right size to hold a specified number of bits of data. The actual form of these storage registers varies with the machine.

Secondary Storage: Slide Switches

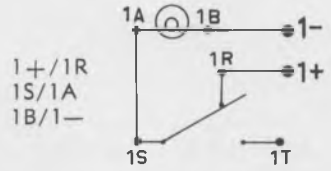
The six slide switches of MINIVAC 601 are used to supply secondary storage. These switches are operated manually and are used to store binary information according to the convention:

switch right = 0
switch left = 1

The following experiment illustrates secondary storage in MINIVAC.

EXPERIMENT 3: SECONDARY STORAGE

This experiment uses MINIVAC 601 to demonstrate how a digital computer remembers information which is not required immediately by storing the information in SECONDARY STORAGE. The slide switches are used as secondary storage for MINIVAC 601. The operator transfers data to the SECONDARY STORAGE by manually operating the slide switches.



1. Turn power ON. Move slide switch to the left. Light 1 comes ON to indicate that a "one" (binary data) is being stored in SECONDARY STORAGE, Section 1. The SECONDARY STORAGE slide switch continues to remember "one" until it is instructed to remember a "zero" by the operator.
2. Move slide switch 1 to the right. Light 1 goes OFF to indicate that a "zero" (binary data) is now being stored in SECONDARY STORAGE, Section 1.

Most large digital computers automatically transfer data at very high speeds to secondary storage. Modern computers may use punched cards, punched paper tape, magnetic tape, and other special devices for secondary storage.
Secondary and external storage are required by digital computers because there is not enough capacity in the storage-processing unit to store all the information required by some problems. Excess information which is not required immediately is transferred to secondary or external storage. External storage for a digital computer is similar to the file cabinet which some people keep in the basement. Some papers may be thrown away, but there is not room enough in the desk upstairs to keep all items which must be filed. "Active" information, which is used frequently, is kept close at hand in the desk and "dead" information, which is used infrequently, is kept in a file cabinet in the basement. Secondary storage in this case would be information required only occasionally, but important enough to be kept close by. This information might be kept in the back of a desk drawer, or perhaps in a cabinet several steps from the desk.

Internal Storage—Relays as Memory

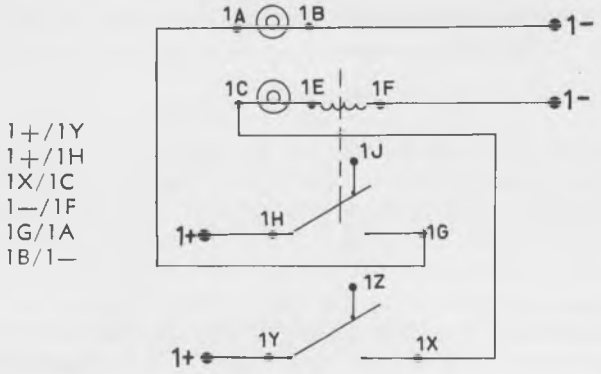
Before going on to the more complicated forms of storage used in large computers, we will turn again to MINIVAC 601 and examine the storage system used to provide memory for this small computer. Storage or "memory" for MINIVAC is supplied by the 6 relays. These relays can be used to store or "remember" six binary bits. As indicated above, a large computer may have storage capacity for 36 binary bits in each of its storage registers. Thus the storage capacity of MINIVAC 601 with its 6-bit register is one-sixth that of a register in the 36-bit machine.

As noted above, in a large computer each storage register is identified by a location number.

In the case of MINIVAC 601 it will not be necessary to consider this problem of memory location identification, since we are dealing with only one storage register.

With the exception of the distinctions noted above, the basic functioning of the storage system in MINIVAC 601 exactly duplicates the functioning of the storage system in a larger computer. Thus, the larger computer may be thought of as simply an extension of many MINIVAC 601's lined up in a row and interconnected. To duplicate the storage capacity of the IBM 7090 computer system, for instance, would require 192,000 MINIVAC 601 Computers in combination.

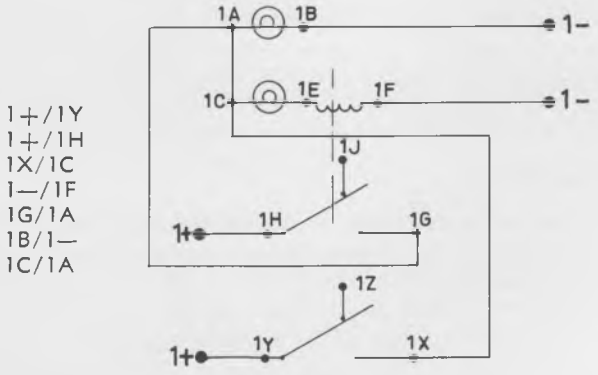
In the discussion of the operation of the relay in Book I, the relay was used as a switch operated by a pushbutton. It was noted that the relay could be used to indicate the two-valued binary code by considering the relay OFF to be storing or remembering a '0' while the relay ON was thought of as remembering '1'. The relay light could be used to indicate when the relay was storing a '1' (light on) or a '0' (light off). This simple circuit may be thought of as a manual memory: as long as the pushbutton is being held DOWN, the relay remembers a one. When the pushbutton is released, the relay ***forgets*** the one and starts remembering a zero. The relay in this circuit is a ***memory element*** controlled by the pushbutton.



MANUAL RELAY MEMORY CIRCUIT

A more efficient memory unit would be achieved if it were possible to make the relay ***remain*** in the DOWN or 1 position once it was signaled to go to this position by the pushbutton. Such a circuit would enable the relay to remember a 1 once it had been signaled by pushing the pushbutton to "forget 0 and start remembering 1".

The manual memory relay circuit above can be easily modified to achieve this by using one of the switches of the relay to continue to supply current to the coil of the relay after the pushbutton has been released. Making use of this relay switch, the pushbutton will initially supply current to the relay. Once current is supplied to the relay coil, the relay will close and a second path for the current will be supplied through the normally open switch of the relay. With this circuit wired, the relay becomes "self-locking" in the 1 position.

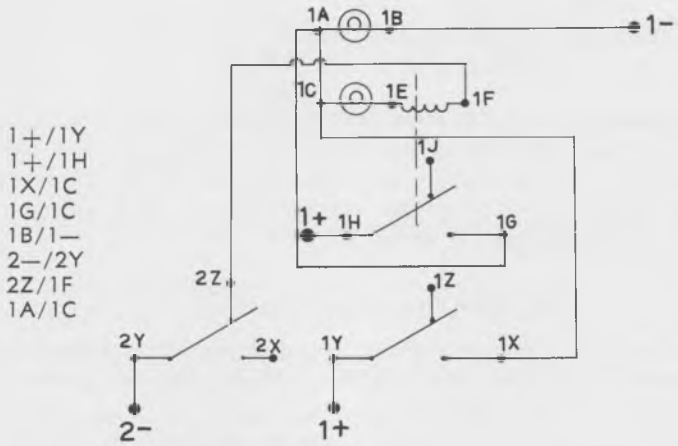


SELF-LOCKING RELAY MEMORY CIRCUIT

The evident difficulty with this self-locking relay memory circuit is that, although the relay will now remember a 1 once it is told to remember 1, it cannot ***forget*** 1 unless the current to the computer is turned off. The next step in building a usable memory unit is obviously to modify the circuit so that the relay can forget 1 and start remembering 0 again.

As an initial step in the direction of programming such a circuit, a second pushbutton may be used to supply a means of disconnecting power from the relay circuit. Using this circuit, pushbutton 1 will be used as in the self-locking memory circuit to supply current to the relay coil and to cause the relay to stop remembering 0 and start remembering 1.

Pushbutton 2 will be used to cause the relay to ***stop*** remembering 1 and ***begin*** remembering 0. As indicated below, this program uses the normally closed contacts of pushbutton 2 and the normally open contacts of pushbutton 1. The self-locking connection used in the previous circuit is retained.



TWO-BUTTON RELAY MEMORY CIRCUIT

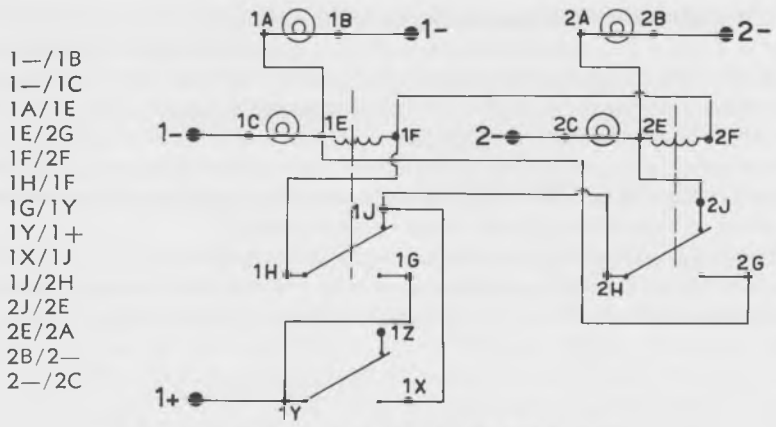
The two-button or "two input" memory circuit provides a workable memory element which satisfies the condition that it be able to remember a 1 ***or*** a 0 on signal. The weakness in this circuit is that a different signal is required to tell the memory circuit to forget 0 and to begin remembering 1 than is used to tell it to forget remembering 1 and recommence remembering 0. Pushbutton 1 serves as the "forget 0 remember 1" signal and pushbutton 2 provides the "forget 1 remember 0" signal.

It would be particularly desirable if we could obtain a memory circuit which would respond to a single signal such that, if the circuit were remembering 1, it would forget 1 and start remembering 0. In short, it would be desirable to have a "single-input memory circuit".

The Single Input "Flip-Flop"

Before programming MINIVAC 601 for a single-input memory circuit a comment on terminology is in order. An examination of the operation of the two-button relay memory circuit will show that as the contents of the memory are changed, the relay goes first ON and then OFF, changing from one position to another. The switching motion of the relay has caused engineers to refer to this type of circuit as a "flip-flop" and the two-button relay memory circuit above is known as a "two-input flip-flop".

So, the circuit which we are now seeking is a "single input flip-flop." The circuit and program below are for a single input flip-flop. Once this circuit has been programmed on MINIVAC 601, pushing ***one*** pushbutton will signal the computer to remember a 1 or a 0. If the relays are remembering a 1, pushing pushbutton 1 will signal them to "forget 1 and start remembering 0". If the relays are remembering a 0, pushing pushbutton 1 will signal them to "stop remembering 0 and start remembering 1". Light 1 will be ON when the flip-flop is remembering a 1, and will be OFF when the flip-flop is remembering a 0.



SINGLE INPUT FLIP-FLOP

In Book IV, the single input flip-flop will be used to provide the memory necessary to enable the computer to count and to perform various mathematical operations. For the purposes of this discussion of computer storage, it is necessary only to note that this single input flip-flop has the characteristics of the basic element of a computer memory system. It is able to remember a 1 or a 0 and to "change its mind" when signaled to do so.

The Basic Processing Function

The detailed operations of the processing unit are discussed in Books III and IV. In this discussion, it is sufficient to note that the processing unit performs two basic kinds of operations.

(1) It ***controls*** the flow of information within the computer. The processing unit "directs traffic" within the computer, sending information from the input unit to storage, from storage to the various calculating units within the processing unit, and from processing to output or storage. In one sense, the processing unit might be thought of as the bookkeeper of the computer. It keeps track of and handles the placement of all data.

(2) It does all calculations. The processing unit is the calculator of the computer. In the processing unit arithmetic data is modified and "decisions" are made. Instructions to the computer directing it to perform operations on data are carried out. All other units of the computer are used to facilitate transmission and storage of information. Only the processing unit actively modifies or uses data. The other units of the computer are passive with respect to the data.

Human Processing

The function of the processing unit can be visualized in terms of the division problem discussed earlier. Once the human operator has the divisor and dividend written on paper and has been given the list of instructions for performing the division, his actions are analogous to those of a processing unit. He will follow the instructions, acting upon his data, until he reaches the final answer.

The processing unit of a computer acts just like the human operator just mentioned ***after*** receiving both instructions and data. In a sense then, the processing unit is the work center of the computer.

MINIVAC 601 Processing

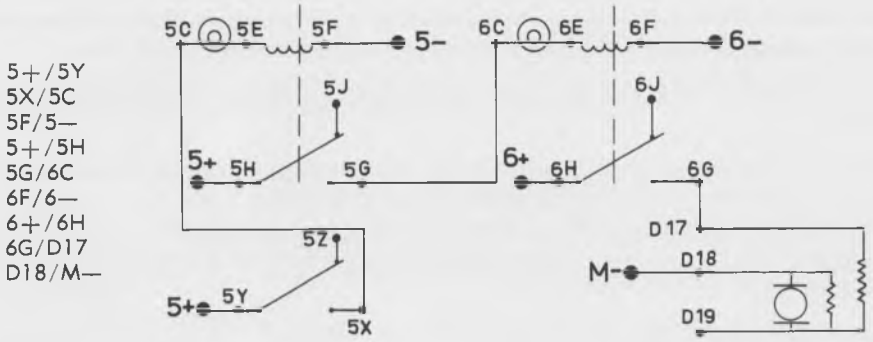
Processing in MINIVAC 601 is accomplished using logical circuits to duplicate arithmetic and decision functions. Relays are used as switching devices and their operation is controlled by instructions communicated to MINIVAC by means of the program wired on the computer console. During the "execution" of a program, the processing section of MINIVAC 601 controls the operation of the computer. The program wired on the computer console indicates to the processing section what operation it is to perform and the processing section controls the other units of the com-

puter—obtaining information when it is required from input sources and communicating the final answer to the output units.

The following experiments illustrate some of the basic functions which a computer can perform, using the processing unit in various ways.

EXPERIMENT 4: CONTROL

This experiment uses MINIVAC 601 to demonstrate how the processing unit ***controls*** the computer's operations. The rotary switch is ***controlled*** by the Processing Unit.



Turn power ON. Push pushbutton 5. This energizes relay 5 (turns it ON) and causes current to flow through the switch contacts of relay 5 to relay 6. Pushbutton 5 ***controls*** relay 5. Relay 5 in turn ***controls*** relay 6. Relay 6 ***controls*** the operation of the motor.

The motor is ***controlled*** by relay 6 which is part of the Processing Unit. Relay 6 is ***controlled*** by relay 5 of the Processing Unit. Relay 5 is controlled by Binary Input pushbutton 5. The operator supplies the ***Input*** by pushing pushbutton 5. The program wired on the computer console gives the instruction:

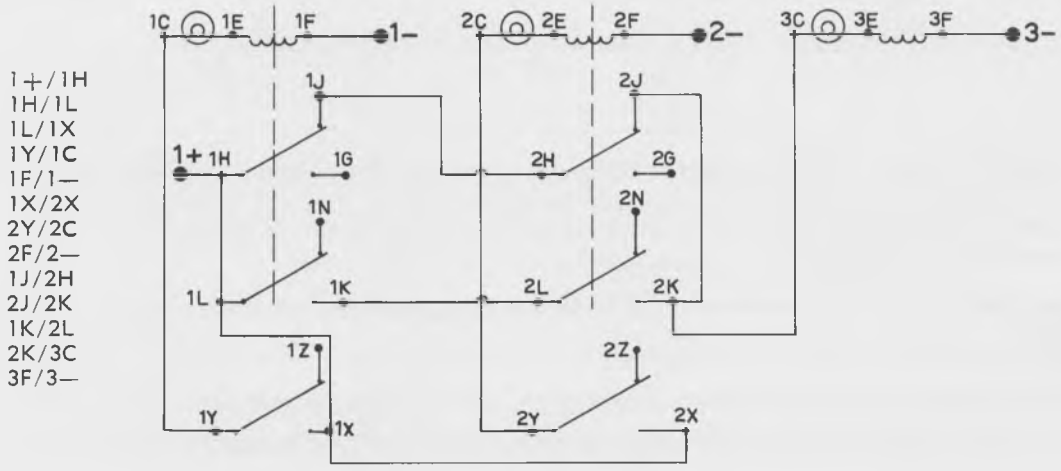
If pushbutton 5 is DOWN, turn relay 5 ON.
If relay 5 is ON, turn relay 6 ON.
If relay 6 is ON, turn motor ON.

In a more complicated program, the relay could be controlled by other relays or switches as the result of some calculation or series of events.

This ability of a digital computer to control its own operations permits the computer to automatically make a whole series of calculations at high speed without direct operator command.

EXPERIMENT 5: DECISION MAKING

This experiment uses MINIVAC 601 to demonstrate how the Processing Unit of a computer can make decisions based on rules given by the programmer. Two binary numbers are compared and the computer ***decides*** whether or not the numbers are equal.



1. Transfer the ***decision rule*** to the computer by wiring the program onto the console. Pushbuttons 1 and 2 will represent the numbers to be compared. Relay 3 will indicate the computer's decision. If the numbers are equal, relay 3 will come ON. If the numbers are NOT equal, relay 3 will go OFF. (Note: a pushbutton UP represents zero; a pushbutton DOWN represents 1.)

2. Turn power ON. Relay 3 comes ON because both numbers are zero.

3. Push pushbutton 1. Relay 3 goes OFF because the numbers are no longer equal. (The first number is now 1; the second number is 0.)

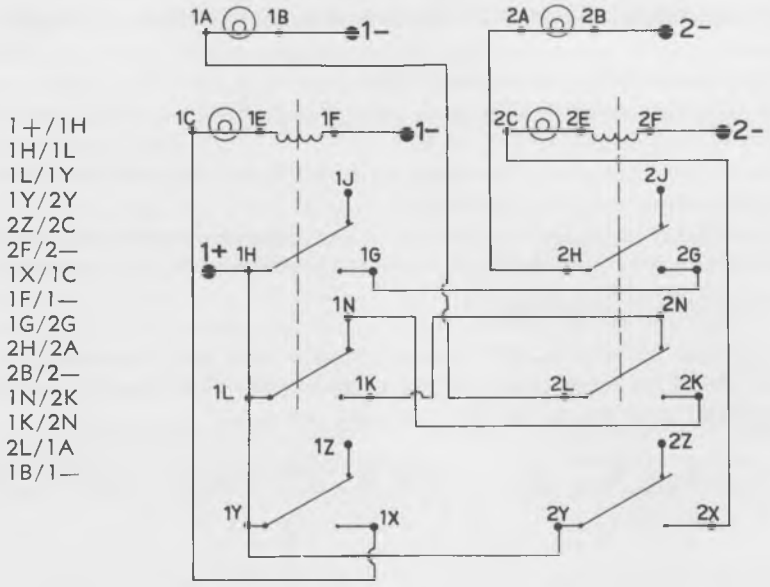
4. Push pushbutton 2 while holding down pushbutton 1. Once again the numbers are equal (both are 1) and relay 3 comes ON.

5. Release pushbutton 1 while holding down pushbutton 2. Relay 3 goes OFF because the numbers are not equal.

This decision rule can be programmed without using relays 1 and 2, in which case the pushbutton ***contacts*** perform the processing function. Since the relays have twice as many contacts as the pushbuttons and can be electrically operated, they are considerably more versatile than the pushbutton contacts. For this reason, relays are used to perform the processing function in all but the simplest programs.

EXPERIMENT 6: ARITHMETIC

This experiment demonstrates how the Processing Unit of a digital computer may be used to perform arithmetic calculations. Two numbers (0 or 1) are added together and the answer is indicated by the Binary Output lights.



1. Transfer the rules of addition to the computer by wiring the program onto the console. Pushbuttons 1 and 2 will represent the numbers to be added. Light 1 ON will represent an answer of 1; light 2 ON will represent an answer of 2; NO lights on will represent an answer of 0.
(Note: a pushbutton UP represents 0; a pushbutton DOWN represents 1)

2. Turn power ON. No lights come on because the input numbers are both zero: $0 + 0 = 0$

3. Push pushbutton 1. Light 1 comes ON: $1 + 0 = 1$

4. Push pushbutton 2 while holding pushbutton 1 down. Light 2 comes ON: $1 + 1 = 2$

5. Release pushbutton 1 while holding pushbutton 2 down. Light 1 comes ON: $0 + 1 = 1$.

The Basic Output Function

The output function is basically the communication of the processing unit's results to the outside world. Through the output unit, information about the problem or the operation of the computer is obtained from the computer.

Human Output Functions

In terms of the previously discussed human analogy, output is obtained from the human being when the answer is written on the paper (output media) by the pencil (output unit) which is controlled by the human being (processing unit). In the case of the human being, the output device is used to communicate both partial information from the human processing unit during computation of the problem and to display final answers upon completion.

MINIVAC 601 Output Units

Output information is obtained from MINIVAC 601 using two output devices: the BINARY OUTPUT lights and the DECIMAL OUTPUT mechanism. Information is "read" from the binary output light by interpreting a light which is on as communicating a "1" and a light which is off as communicating a "0".

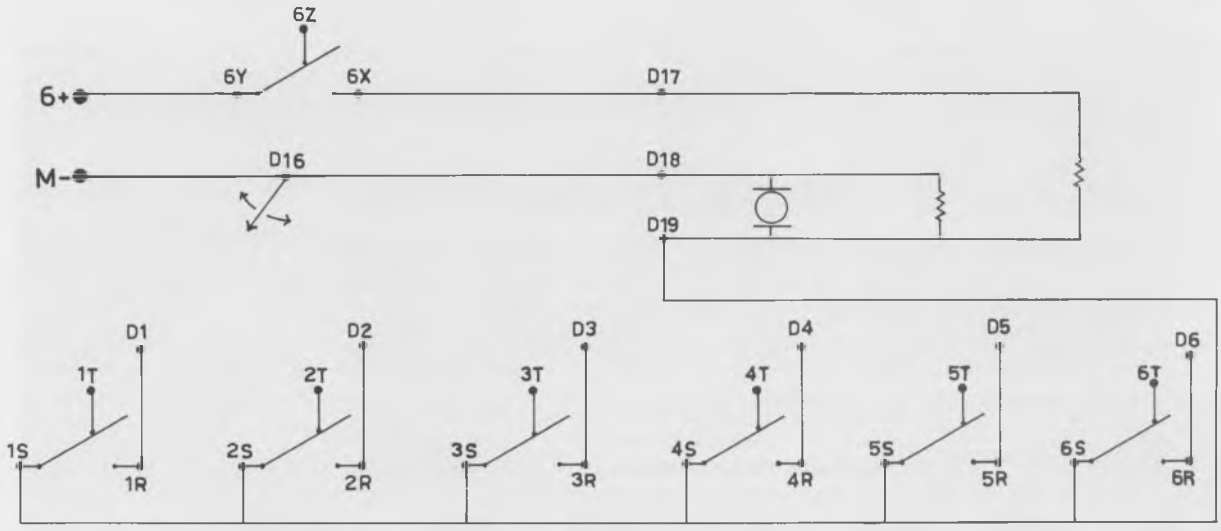
Output information is "read" from the decimal output mechanism of MINIVAC 601 by noting the number at which the pointer knob is pointing.

An example of Binary Output is given in experiment 6. The Binary Output lights were used to indicate the answer to a simple addition problem.

The following experiment illustrates the use of the rotary switch for Decimal Output.

EXPERIMENT 7: DECIMAL OUTPUT

This experiment demonstrates how a computer delivers decimal output information to the operator. The rotary switch (Decimal Output) is controlled through the slide switches (Secondary Storage) to communicate to the operator the contents of Secondary Storage.



1R/D1	3S/4S	6R/D6	6Y/6+
1S/2S	4R/D4	6S/D19	
2R/D2	4S/5S	D18/D16	
2S/3S	5R/D5	D16/M—	
3R/D3	5S/6S	D17/6X	

1. Set all slide switches in the RIGHT position. The slide switches will represent the numbers 1 through 6. To place a number in secondary storage, the appropriate slide switch will be

moved to the LEFT position. Pushbutton 6 will be used to give the computer the instruction:
Indicate number in Secondary Storage using Decimal Output rotary switch.

2. Turn power ON. Move slide switch 4 to the LEFT position. This places the number 4 in Secondary Storage.

3. Push pushbutton 6. This instructs the computer to communicate the contents of Secondary Storage to the operator. The rotary switch turns to 4 on the Decimal Input-Output dial. The number 4 is the Decimal Output.

4. Move slide switch 4 to the RIGHT position and select another number to be placed in Secondary Storage. Again, push pushbutton 6 to instruct the computer to communicate the contents of Secondary Storage through Decimal Output.

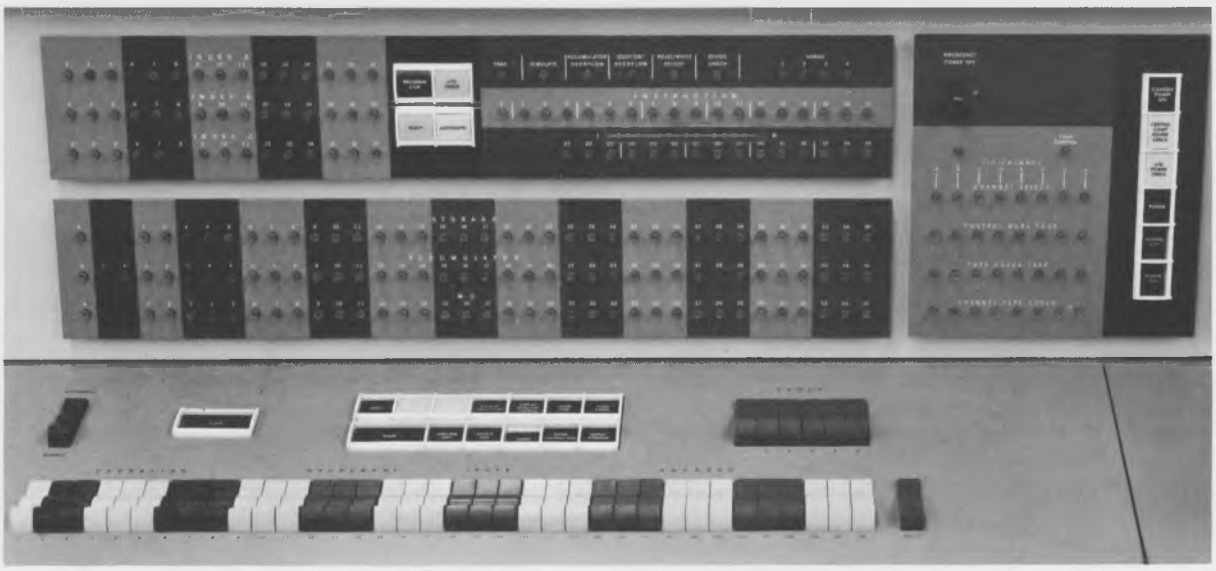
3. EXPANSION OF THE BASIC COMPUTER FUNCTIONS

Input Media and Codes

Limited amounts of information can be communicated directly to a large computer through its console. This direct input information may be either binary or decimal in form, depending upon the operating characteristics of the machine, and is normally communicated through switches on the main console of the computer. In some situations direct communication is obtained through the use of a "flexowriter" which is a typewriter-like device which, in addition to printing characters on a written page, transmits a coded representation of the characters to the computer for interpretation.

Direct Console Input

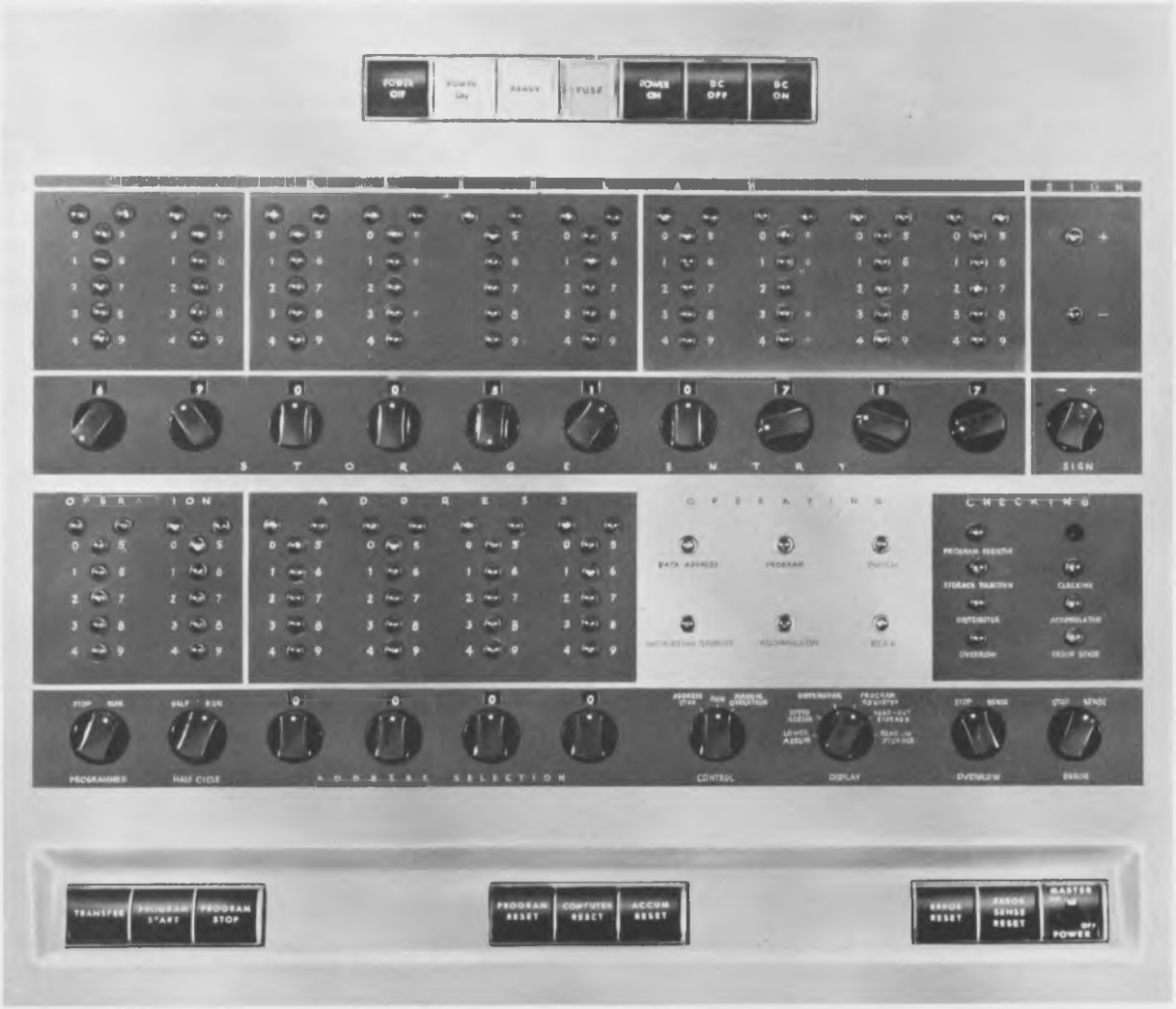
In the photograph of the 7090 console below the input buttons used to provide direct binary input information to the IBM 7090 computer are visible. The functioning of these buttons is comparable to the operation of the binary input buttons on the MINIVAC 601 Console. When a particular button is pressed down, it represents a one in that position and when the button is in the normal or up position, it represents a zero in that position.



IBM 7090 CONSOLE

In the photograph of the IBM 650 Computer console below, the input switches which communicate input information in decimal rather than binary form are indicated. The operation of

these switches is similar to the operation of the decimal input switch on the MINIVAC 601 Console. To communicate a decimal number to the IBM 650, the appropriate switches are turned so that the desired decimal number is indicated by the switches.



IBM 650 CONSOLE

In addition to direct console input in the binary and decimal form using the same coding system employed with the MINIVAC 601, large commercial computers have several other forms of input. Basic input media in addition to direct console input are: punched cards, punched tape, and magnetic tape.

Punched Card Input

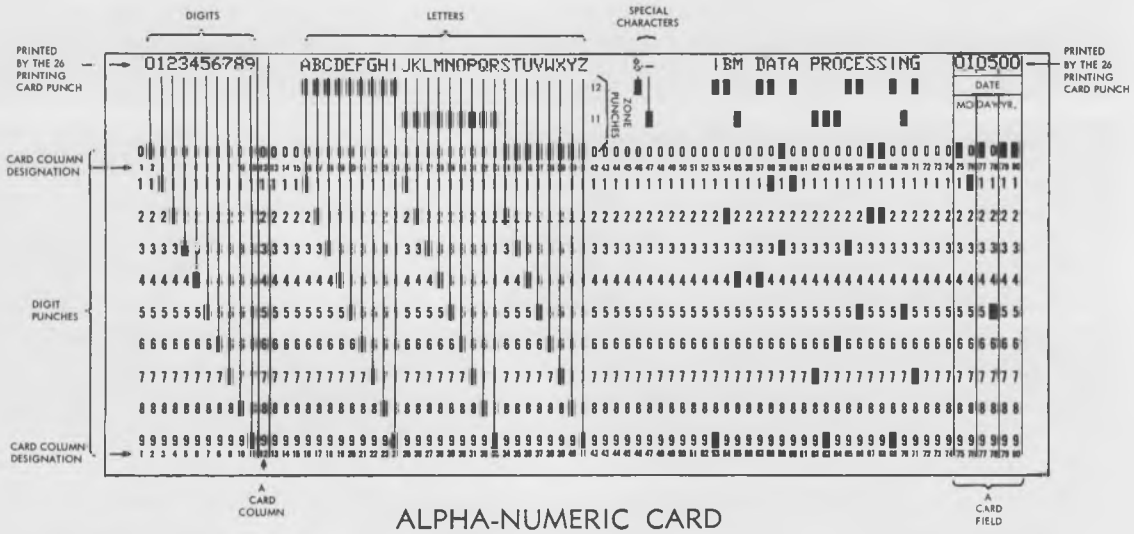
Punched cards are perhaps the most common medium for communication with a computer. Information is "recorded" on the cards by means of a small hole punched in a particular location on a card in accordance with a coding plan which the machine has been programmed to "understand". As the machine "reads" a card, it obtains information by sensing the presence or absence of holes in each of a number of locations. The information obtained from the reading of the card is then translated into electronic information for processing and storage within the computer.

Alpha-Numeric Card Code

The standard alpha-numeric code is summarized in the photograph below. The numbers zero through nine are coded as a single punch in a vertical column. The alphabetic characters and symbols are represented by two punches in a single vertical column.

Communication using the standard alpha-numeric input card code is limited to 80 characters (numbers, letters, or special characters) per card. The use of "binary" codes greatly increases the amount of information which may be communicated using a single card.

As noted earlier, although the punch itself is a binary variable—a variable which can have only two values, either 0 or 1—the use of the binary punch in conjunction with the alpha-numeric code creates a situation where only one variable value may be communicated in each vertical column. In a sense, this is similar to the limitation imposed by the decimal input dial on the MINIVAC 601. Although at any location the pointer is either pointing at the number or is not pointing at that number (corresponding to the punch either being in a particular location or not being in that location) meaningful information can only be communicated by considering all 16 locations in which the pointer ***might*** be. This corresponds to considering all 12 positions in a given vertical column on a card in which a punch ***might*** be located.



ALPHA-NUMERIC CARD

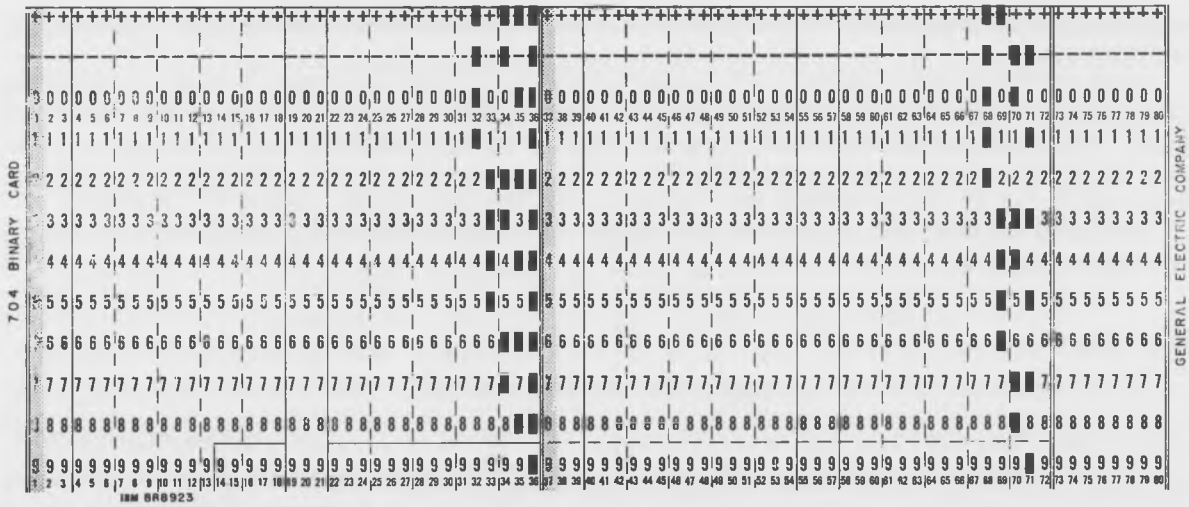
Binary Card Codes

As was noted earlier, the binary number system is discussed in detail at the beginning of Book IV and, for our purposes at this time, it is necessary only to remember that a binary code is one made up of only two characters (0 and 1). Since the binary code is made up of only zeros and ones it is unnecessary to consider the location of the punch in a vertical column in order to know the ***value*** of the ***binary digit*** communicated by the punch. The presence of a punch communicates a one and where there is no punch a zero is communicated. Since there can be ***only*** zeros and ones in a binary code ***all*** information which could be communicated about the value of a particular binary digit is communicated simply by the presence or absence of a hole in a particular location on the card.

The length of a binary number or "word" (i.e., the number of binary characters in the number or word) can vary. For example, the number labeled (A) below is a four-digit binary number or word and the number labeled (B) is a ten-digit binary number or word.

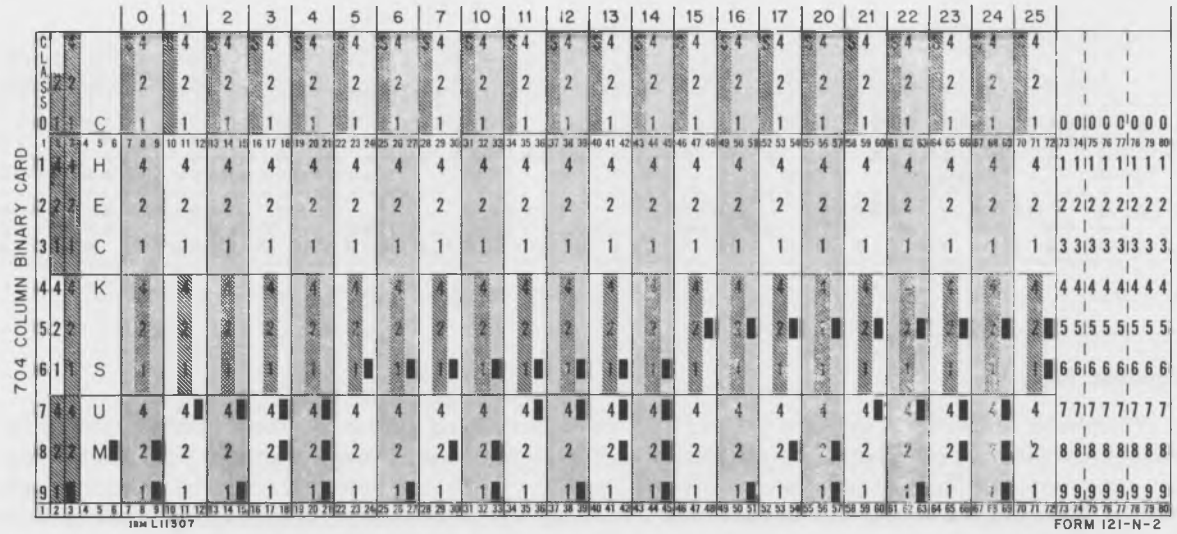
(A) 1010
(B) 0000001010

The card below is an input card prepared using a binary code. On this card binary information is represented in twenty-four 36-digit words occupying positions in columns 1 through 36 and 37 through 72 respectively in 12 ***rows***. This is a "row binary" card.



ROW BINARY CARD

In place of punching binary information in horizontal rows as in the row binary card code, information may be punched along the ***vertical*** columns of the card. Twelve binary digits can be punched in each vertical column. Thus, in the case of a 36-digit word length, ***three*** vertical columns are required using the column binary code to communicate the information contained on ***one*** above in row binary form.



COLUMN BINARY CARD

The Binary Nature of Input

It is important to note that all input information is read from a card as 36-digit binary words. Once the information has been read into the computer, it can be ***interpreted*** by the computer in accordance with any code that the computer has been programmed to understand. Thus, standard alpha-numeric code cards, row binary cards and column binary cards are all read into the computer by the same process. However, once the information is in the computer, it is interpreted by the program prepared for the computer so that the computer can "understand" the information communicated according to the appropriate code on the input card.

The important point to remember in this context is that ***all communication*** to the computer ***is*** actually ***binary communication.*** Any non-binary form of information is read by the computer ***as if*** it

were binary information and then interpreted "in binary" so that the computer can "understand" the information.

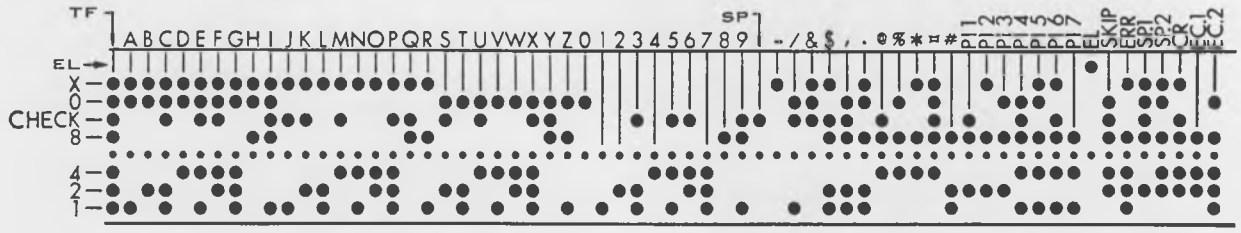
Since MINIVAC 601 operates in the same manner as a large computer, it must also be given information in binary form. This is the reason for the six binary input buttons on the MINIVAC console. When non-binary information is to be supplied to the computer as, for example, when using the decimal input dial, MINIVAC 601 will be programmed to interpret this non-binary information and to store and process the non-binary information in binary form.

In Book IV a program is given for conversion from decimal to binary. (See Decimal to Binary Converter—Book IV.)

Paper Tape Input

In a manner similar to that in which information is communicated on a punched card, information on paper tape is communicated by punching holes in predetermined locations according to various codes. The machine reading a paper tape, as the machine reading a punched card, interprets the presence or absence of a hole in a particular location as binary information. Using paper tape it is possible to communicate information made up of any number of characters to the machine as one continuous record.

The photograph below illustrates the use of the "8 channel" code, one of several coding systems used to communicate information on paper tape. This system is analogous to the alpha-numeric code described above for use with the punched card. The nature of this code can be easily determined by examining the photograph. It should be noted that the smaller dots appearing along the center of the paper tape are analogous to the sprocket holes on the edge of movie film. These small holes are used to move the paper tape past the reading point in the paper tape-reading machine.



EIGHT CHANNEL PAPER TAPE CODE

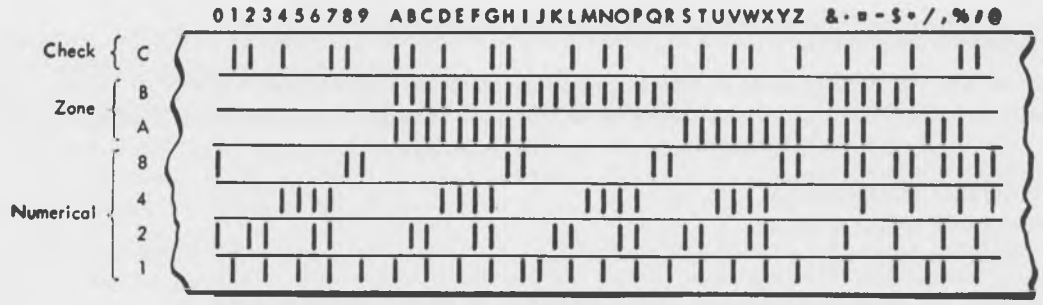
Magnetic Tape Input

Magnetic Tape input systems differ from those discussed above in that information is recorded on a plastic recording tape similar to that used in home tape recorders. In the case of magnetic tape, information is coded as magnetic "marks" on the tape rather than as holes in the tape. A major advantage of magnetic tape is that the magnetic information recorded on the tape can be erased and the tape can be re-recorded and used many times.

A second advantage of the magnetic tape is that the magnetic tape can be read at a much higher speed than the paper tape in a much smaller physical space than is required using either the punched card or the paper tape. Specifically, the same amount of information (3,336 characters) can be recorded on forty-two punched cards or on one inch of magnetic tape (high density).

The photograph below illustrates the use of a seven-bit alpha-numeric code recorded on magnetic tape. Using this code, a character is represented by the presence or absence of magnetic marks in specified positions across the width of the tape.

Binary information may also be recorded on magnetic tape using a 36 bit code similar to that discussed for use in connection with the punched card. The 36-bit word on magnetic tape is recorded in six consecutive columns on the tape. Although seven positions are available on the tape, one position is reserved for information used in checking the reliability of the reading and writing operation.

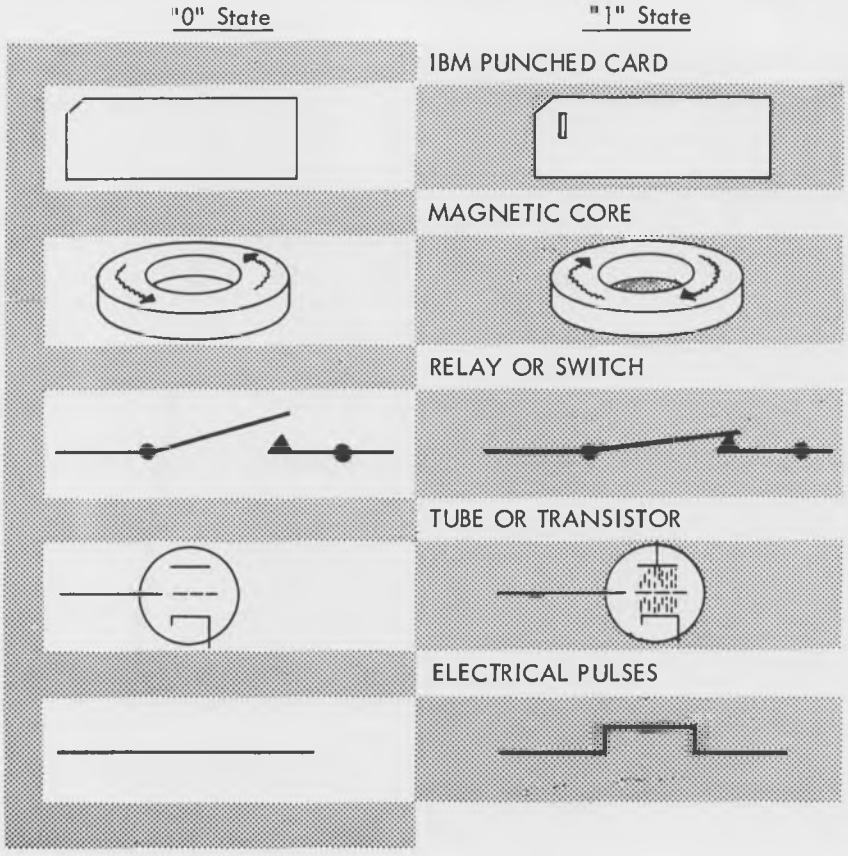


SEVEN-BIT ALPHAMERIC MAGNETIC TAPE CODES

STORAGE MEDIA AND CODES

"Bi-Stable" Elements

In the first section of this book we examined the memory element of the MINIVAC 601. Now we are ready to examine the way in which the same function is performed by different elements in larger computers. In every case, the functioning of the memory element in a large computer is identical to the functioning of a relay serving as a memory element in the MINIVAC 601. The memory element is, in every case, capable of maintaining one of two positions until it is signaled to change positions. Such a two-valued element which can remain in either of two positions until an action outside the element causes it to change to the other position is called a "bi-stable" device. It is a device which can stay (is stable) in either of two positions (the prefix "bi-" indicates two). The photograph below illustrates several bi-stable devices discussed in this book.



BI-STABLE ELEMENTS

Large Computer Storage

The actual form of bi-stable elements which make up the storage register in large computers varies with the machine. However, three basic types of storage are presently in popular use. These are:

magnetic core storage
magnetic drum storage
magnetic disc storage.

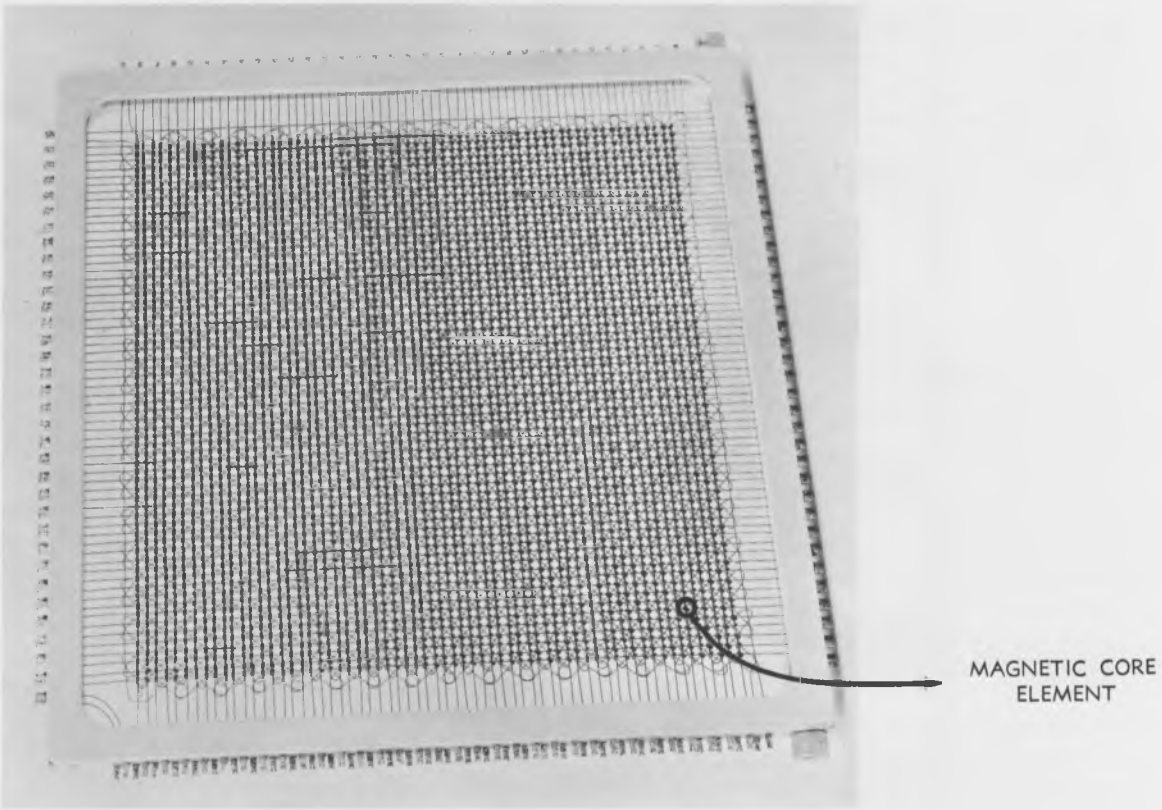
Magnetic Core Storage

The elements of magnetic core storage are small, donut-shaped rings of ferro-magnetic material. These small memory elements, one hundredth of an inch in diameter, can be magnetized in a few millionths of a second and will retain their magnetism indefinitely.

By passing current through a wire going through the center of the donut-shaped ring, it is possible to magnetize the magnetic core memory element. The direction of the magnetic field set up within the core is determined by the the ***direction*** of current flow through the wire. The magnetic core is thus a bi-stable element having two states of polarization representing a 0 and a 1.

Just as in the MINIVAC several relays may be used together to represent a ***series*** of zeros and ones, so in a computer using magnetic core memory, series of magnetic core elements may be combined to create a binary word. A 36-bit binary word is thus remembered by 36 separate magnetic core elements, each of which may be magnetized in either direction.

When using the MINIVAC relay memory element, the "state" or position of the relay was "read" on the light attached to the relay switch. The light ON indicated that the relay was remembering a one and the light OFF indicated that the relay was remembering a zero. The light was used to "sense" the position of the relay.



MAGNETIC CORE PLANE

Reading of the contents of a magnetic core element is accomplished by a "sensing" process. One such sensing process works as follows:

> The magnetic core element is ***forced*** into the zero direction by a current pulse. If the element is storing a one, the ***change*** in direction of polarization induces a pulse in a "sensing wire." If the element is storing a zero, there is no change in the direction of polarization, so there is no pulse.

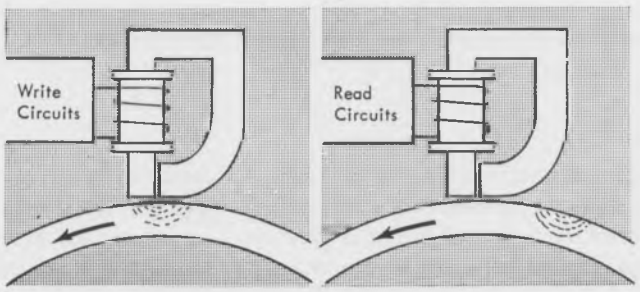
This reading process, however, "destroys" the information content of the magnetic core element. If the information must be retained after reading, a special device which will replace the information as it is read must be used.

The preceding photograph shows a matrix made up of many thousand magnetic core elements which together remember hundreds of thousands of binary bits in a large computer.

Magnetic Drum Storage

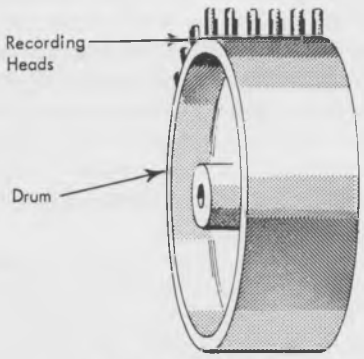
Although the magnetic core provides a much faster means of storing information, some computers use a magnetic drum for storage. The magnetic drum is a steel cylinder coated with material similar to that used on magnetic recording tape. This material can be magnetized with a number of small magnetic "spots" in much the same manner in which information is recorded on magnetic tape. The drum rotates at a constant speed and information is "written on" or "read from" the magnetic spots by a recording head in much the same manner that information is recorded or played back from a tape recording.

In the magnetic drum there is a problem of identifying the location on the drum which is to be read in order to obtain the information desired at a particular point in time. Through a sequencing system, each storage location on the drum is specified by a given address (so that a particular location may be determined) in order to obtain or store information at that point on the magnetic drum.



Writing on a Drum

Reading from a Drum



Magnetic Drum Recording Heads

Because transmission of information can occur ***only*** when a particular location is passing under the recording or play back head, the time required to obtain information from the magnetic drum or to record information in a particular location on the drum is greater than that required to obtain information from the magnetic core.

Magnetic Disc Storage

The magnetic disc uses a process similar to that used with the magnetic drum. Information is recorded on discs which look much like standard phonograph records. The discs are stacked in an array surrounding a central spindle in a mechanism which operates very much like an ordinary juke box.

Information is read into or out of the discs by means of an arm which is able to enter into the stack of records and read from or write on either side of a particular disc. It is important to remember that the recording on the disc is a ***magnetic*** recording. So that although these records appear to be much like the ordinary phonograph records, the recording process is one involving magnetic spots.

Other Forms of Storage

Although the various forms of storage described above are those most commonly encountered in commercial computers, other forms of storage are used to a limited degree.

Binary information is sometimes stored in an electric capacitor with the polarization of the capacitor determining whether the bit stored is a one or a zero. In magnetic recording, the presence of a magnetic spot was taken to indicate a 1and the absence a 0; in the capacitor the presence of extra electrons is taken to indicate a 1 and the absence of electrons a 0.

Cathode ray tubes are also used for storage. When these devices are used, a ***point*** on the screen of the tube is used as the memory element. A charged point represents a 1 and the absence of a charge indicates a 0.

When acoustic delay lines are used as storage elements the binary state of the element is determined by the presence or absence of an ultrasonic vibration in a fluid. The presence of the vibration indicates a 1, its absence a 0.

External Storage

The storage media discussed above are all used for internal and secondary storage. External storage involves media associated with the output units and will be discussed in the next section. At this time it is sufficient to note that three major types of external storage are encountered in the modern digital computer system. These are magnetic tape, punched cards and paper tape.

PROCESSING TECHNIQUES

Large Computer Processing

Processing within a high-speed digital computer is accomplished using electronic circuits analogous with those used in the MINIVAC 601. With some large scale computers the instructions are communicated using a program board in which instructions are wired in a manner similar to the wired programming of the MINIVAC. In most high-speed digital computers, various operations which can be performed by the processing unit are ***permanently*** wired into the computer and the computer is programmed to perform a particular operation (to choose a particular wired circuit) when a specific code is given as an instruction. A computer, might for example, be programmed to choose an addition circuit when it encounters the instruction "01" and to choose a subtraction circuit when it encounters the instruction "02."

When coded instructions are used in a large digital computer, the numbers representing the coded instruction may be stored in the storage unit of the computer just as data is stored. When this is done, the processing unit is directed to certain registers of the storage unit to obtain numbers which are interpreted as instructions and to other registers to obtain numbers which are interpreted as data. When this process is followed, the computer is said to be operating under control

of a "stored program." In one sense then, the MINIVAC 601 or any other wire-programmed computer may be said to be operating under control of a stored program. In the case of the numerically coded instruction machine, the program is stored as numbers in the storage unit while in the case of the wire-programmed machine the instructions are stored in the connections wired on the programming panel of the computer.

Electronic circuits used in the high-speed digital computers use components which are different in form from those used on the MINIVAC 601. The ***functions*** which they perform, however, are similar to the operations of the relays and rotary switch of the MINIVAC 601, and the MINIVAC components offer the particular advantage of being completely visible so that their operation can be easily viewed. It is easy to see the switching action of a relay on the MINIVAC 601 but impossible to see the similar functioning of a transistor or magnetic core element.

The processing unit of the computer is the central control instrument of the computer system. The processing unit is usually associated with the "main frame" of the computer and direct communication with the processing unit is provided for the operator through the switches and lights of the computer console.

The Binary Nature of Processing

The processing unit of most large computers operates only in binary. Just as the relays used for processing on the MINIVAC are capable of dealing only with zeros and ones, the processing systems of larger computers work only in binary code.

Since information is communicated to the machine in other than binary form, it must be coded in binary form according to a system which will permit the machine to recognize it as numeric, alphabetic, or special character information. This binary coding of non-binary information is often accomplished using a Binary Coded Decimal or "BCD" code rather than using the binary equivalents of the decimal number (see first section of Book IV). The following chart provides a summary of the representation of alphabetic, numeric and special characters according to the ***Binary Coded Decimal System.***

THE BINARY CODED DECIMAL SYSTEM USED FOR PROCESSING

Character	BCD	Code	Char.	BCD	Code	Char.	BCD	Code	Char.	BCD	Code
blank	110	000	A	010	001	N	100	101	+	010	000
1	000	001	B	010	010	O	100	110	—	100	000
2	000	010	C	010	011	P	100	111	/	110	001
3	000	011	D	010	100	Q	101	000	=	001	011
4	000	100	E	010	101	R	101	001	-	001	100
5	000	101	F	010	110	S	110	010	.	011	011
6	000	110	G	010	111	T	110	011	)	011	100
7	000	111	H	011	000	U	110	100	$	101	011
8	001	000	I	011	001	V	110	101	*	101	100
9	001	001	J	100	001	W	110	110	,	111	011
0	000	000	K	100	010	X	110	111	(	111	100
+0	011	010	L	100	011	Y	111	000			
—0	101	010	M	100	100	Z	111	001			

In Book III of this series the decision-making function of the processing unit is discussed, and examples of the operation of the processing unit as a decision-making device are provided. Book III also contains a detailed discussion of the logical operations which enable the processing unit to function as a "Reasoning" device. In Book III you will discover how MINIVAC 601 is able to duplicate many of the functions of human reason which are normally considered proof of a human being's ability to "think."

OUTPUT MEDIA AND CODES

Direct Output

Just as binary information may be given directly to the computer using input buttons on the computer console, binary output may be obtained directly from the computer through lights similar to the binary output lights of the MINIVAC 601. When binary output lights are used, the

code already established for use with the MINIVAC 601 is employed. A light ON indicates a "1"; a light OFF indicates a "0."

A photograph below shows the direct output lights on the console of the IBM 7090 computer. You will notice that there are 37 lamps in the row of output lights of the 7090 computer. These are used to display the contents of the 36 binary "bits" in a 7090 storage register. The far left light indicates the sign of the number when the lights are displaying a binary number. The "sign indicator" light ON represents minus; the light OFF represents plus.

THE IBM 7090 CONSOLE

Large Computer Output Units

Output information may be obtained from a high-speed digital computer using lights and physical indicators similar to the binary output lights and decimal output dial of the MINIVAC. When output devices of this type are employed, they are generally used only to obtain limited operating data from the computer and are usually located on the computer console.

More extensive output information is normally obtained using a "printer" unit in which a typewriter-like device produces typewritten pages of numbers or other written information. The printer may be connected directly to the processing unit of the computer in which case information is communicated directly from the processing unit to the printer. The printer may also be connected to a unit which is able to "read" one of the other output media (for example magnetic tape or punched cards). In this second case, output information for printing is actually transmitted from the computer on another media through another output unit and then transformed into printed material away from the main computer.

The discussion of coding and media undertaken while discussing input units is also applicable to the output situation. In essence the output device is simply a machine which reverses the process of the input device. In the case of magnetic tape, the input and output devices are actually in the same physical unit. The same equipment is used to record and to play back the information on magnetic tape.

The unit used to reverse the process of the card reader is the card punch unit and the paper tape reader has a complementary paper punch unit. With both the paper tape and card punch,

a series of dies are used to punch holes in the tape or card in the location corresponding to coded information sent to the punch unit from the processing unit of the computer.

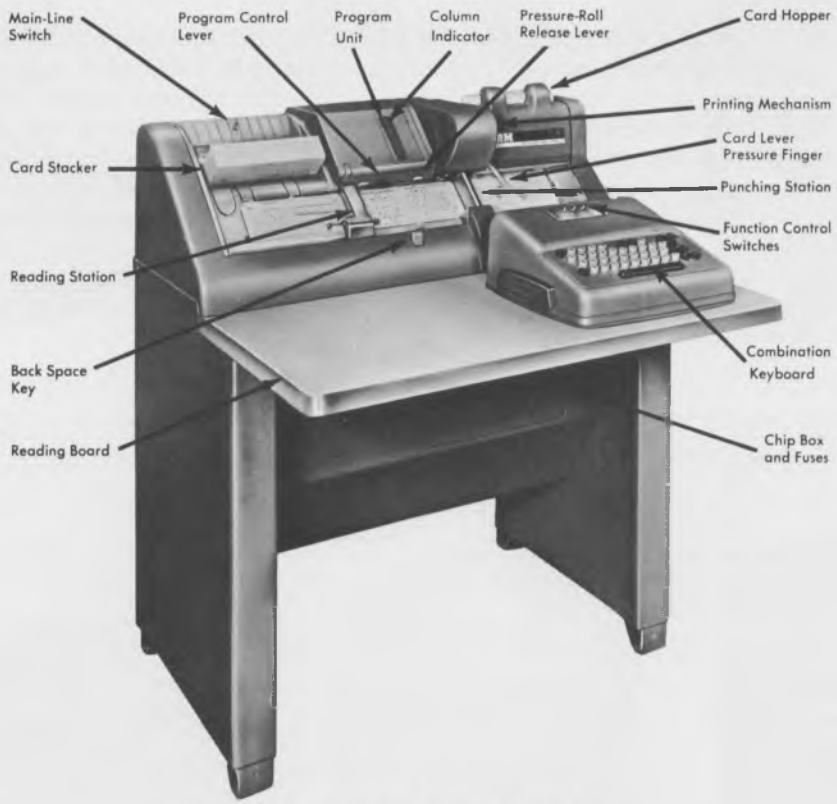
The cathode ray tube output unit which is used to display output in a form which has no counterpart in the input sequence should be noted. This television-like display unit provides an "analog" in the form of a pictorial representation of information. A photograph of a Cathode Ray Tube output unit appears in the next section of this book.

4. COMMERCIAL COMPUTER EQUIPMENT

This section of the book contains photographs of equipment manufactured by the International Business Machine Corporation to perform each of the four major computer functions. These photographs are included to enable you to become familiar with the appearance of units performing the various functions which we have discussed.

Card Input Equipment

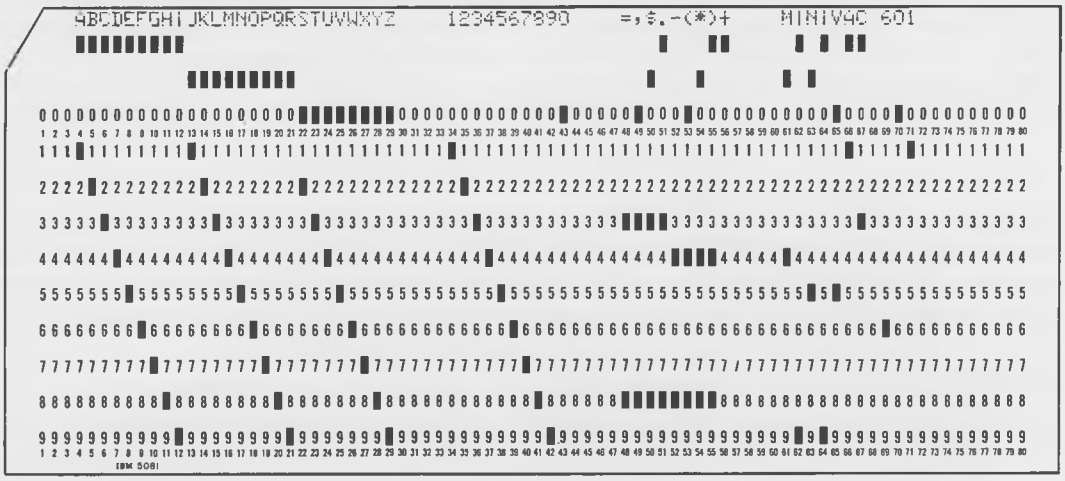
Input information for use by a computer system may be prepared on a card punch of the type illustrated below. The machine in the photograph is the IBM 026 Card Punch which punches cards using the IBM Alpha-numeric Input Card Code.



"IBM-026 CARD PUNCH"

The cards to be punched are placed in the hopper on the right hand side of the 026 and fed through the machine to the punching station. The operator, using the keyboard of the machine, punches the card in much the same way that he would type a letter using a typewriter. The space bar at the bottom of the keyboard is used to move the card under the punch dies without punching. When a key is pressed, a hole is punched in the card in the particular column under the punch dies at the time the key is pushed. The 026 also prints above the column which has been punched the

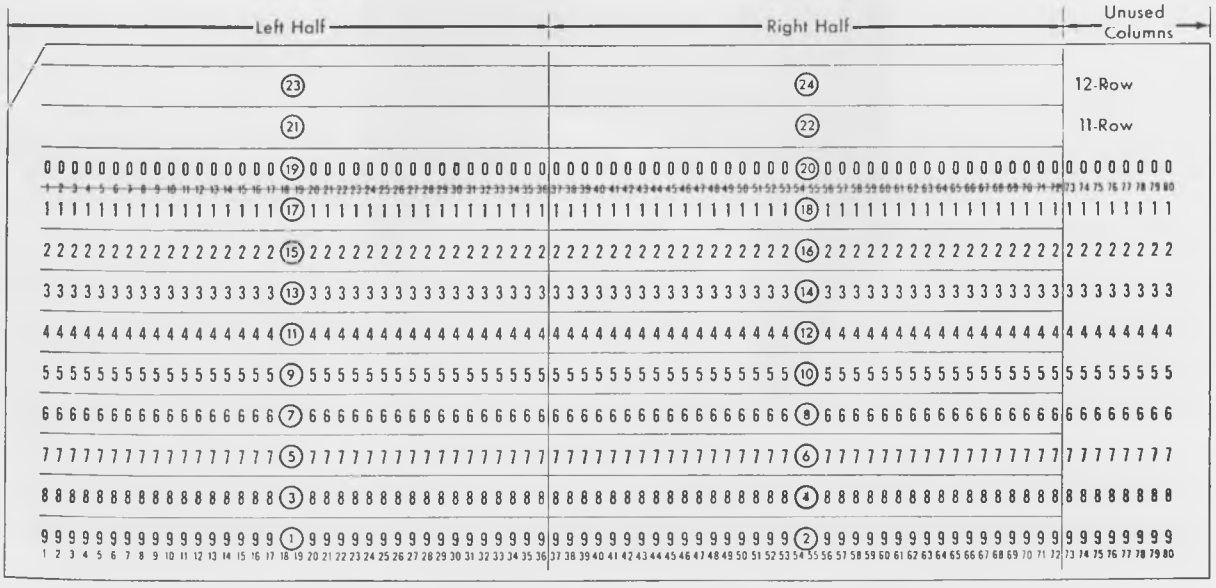
number or letter represented by the punches. An example of a card prepared by an 026 card punch is shown below.



INPUT CARD PREPARED ON THE 026 CARD PUNCH

Once the input information has been punched onto the card, the card can be read by the ***card reader*** of the computer. The photograph below illustrates the IBM 7500 card reader used in conjunction with a high-speed digital computer. The mechanism of the card reader includes two sets of "reading" brushes. Information from the card is read beginning with the first 36 locations in the "nine" or bottom row of the card. After the first 36 locations of the "nine" row have been read, the next 36 locations are read. The last eight locations (columns 73-80) are not normally used for input information but are reserved for sequencing information used to identify a particular card.

For a 72 column read cycle, each corresponding row starting with the "nine" (bottom) row is divided into two 36 column "words". Thus the machine reads twenty-four 36-digit binary words in the order indicated in the illustration below.



THE SEQUENCE FOLLOWED WHEN READING 72 COLUMNS OF INFORMATION FROM AN INPUT CARD

Input-Output Units

The photographs below illustrate IBM card, tape, and printer units.

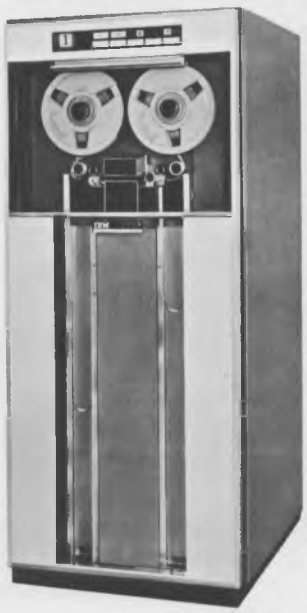
When cards are read by the card reader, they are placed in the hopper at the front of the reader. During the read cycle they pass under the read brushes and out into the hopper on top of the unit.

The magnetic tape unit both reads and writes on magnetic tape. This unit is thus a combined input-output device.

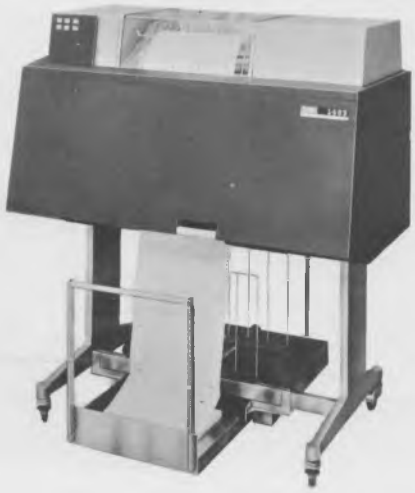
The equipment used to read paper tape operates, as does the card reading equipment, by interpreting the holes in the paper tape as binary information. Once this information is read it is interpreted according to a program previously supplied to the machine. As in the punched cards, a hole is read as a one and the absence of a hole as a zero.

IBM 780 CRT Display

IBM 962 Tape Punch



IBM 729 IV Magnetic Tape Unit



IBM 1403 Printer

IBM 382 Paper Tape Reader

IBM 7500 Card Reader

IBM 7550 Card Punch

Storage Units

Units employing magnetic core, magnetic drum and magnetic disc storage media are illustrated below.

CORE
STORAGE UNIT

DRUM
STORAGE UNIT

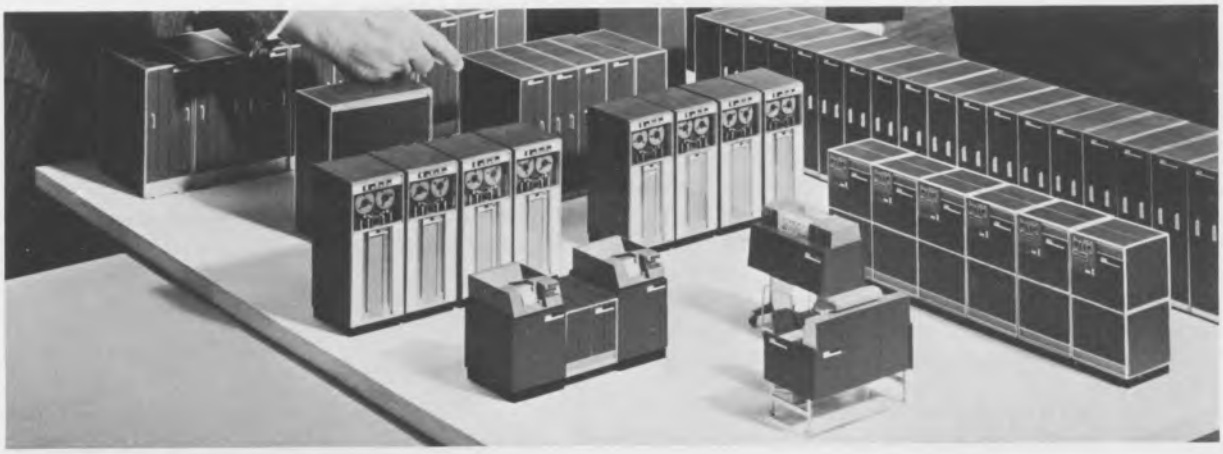
DISC
STORAGE UNIT

5. COMPUTERS OF TOMORROW

Although computer technology has already reached levels of performance beyond imagination only a decade ago, technological progress in the coming years promises to open new horizons for these amazing machines.

The capabilities of these machines of tomorrow can be forecast from the capacities of the most advanced computer systems being operated today. An example is the IBM "Stretch System" illustrated below. The "Stretch" has an internal memory capacity of more than 16 million binary bits. The speed of this system is such that it requires only 2 micro-seconds for "Stretch" to add two numbers and store the results.

The research laboratories of computer manufacturers throughout the country are working now on computer systems with greater capacity, greater speed and more efficient design. Every day, the men and women involved in this new and exciting field put computers to work on new problems, in different areas. The development of the computer is making possible the efficient handling of vast amounts of data and the solution of problems in science, engineering and business.



IBM 7030 STRETCH SYSTEM

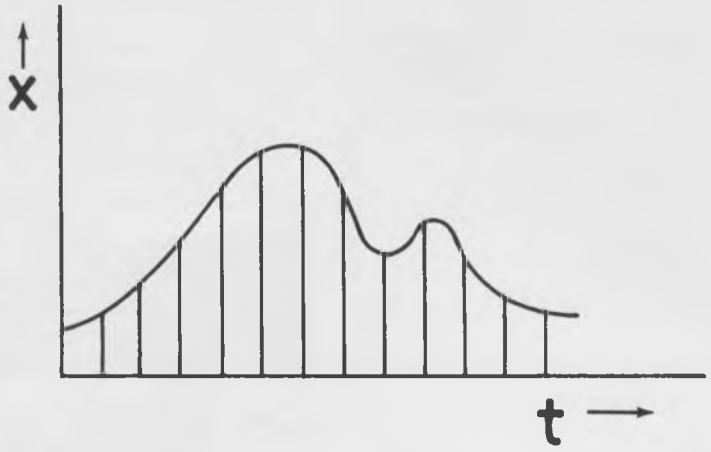
APPENDIX

DIGITAL AND ANALOGUE COMPUTERS

Throughout this manual, all discussion of computers refers to ***digital*** computers. There are in use, however, many computers whose basis of operation is completely different. These are ***analogue*** computers. Both types of computers are capable of performing similar types of operations and can serve similar functions.

The difference between ***digital*** and ***analogue*** computer systems is basically a difference in the method the system uses to handle information. The analogue computer handles information as a continuously varying signal while the digital computer handles information in discreet form.

The following sketch illustrates graphically the difference between continuous and discreet representation:



The solid line represents the value of a function as it changes over time. This is a continuous function; that is, there is a value for the function at ***every*** point in time. This representation can be handled by an analogue machine.

The vertical bars also represent the value of the function as it changes over time. However, the function exists only at the distinct points in time represented by the bars; the function is not defined between these points. As the points in time are chosen closer and closer together, a line connecting the ends of the bars will approach the continuous curve.

The reason for this difference in form of information is essentially because of a difference in the type of components which go into the systems. The analogue computer uses mechanical or electrical components to represent relationships. An analogue computer might, for example use a rotating shaft with various shaft positions corresponding to values of a variable. Or, an analogue computer could use an electric capacitor—which is capable of storing electrons—with various levels of charge on the capacitor corresponding to values of a variable.

The digital computer, on the other hand, uses various components to represent specific values of a variable. The digital computer can, for example, use the two positions of a relay (on or off) to represent two distinct values of the variable.

As an example of how the two systems differ in operation, consider the problem of adding two pounds of salt plus three pounds of salt:

To solve this problem as an analogue machine would, we would do the following:

1. Set a scale to read zero
2. Pour 2 pounds of salt onto the scale (the scale indicator would climb as the salt was being poured onto it).
3. Pour 3 more pounds of salt onto the scale (once again, the scale indicator would climb as the salt was being poured onto it).
4. For the answer, we would read the final result on the scale: $2 + 3 = 5$.

If we wished to do this on an analogue machine, we would let amperes of current represent pounds of salt. That is, we would use numbers of amps of current as ***analogues*** of the numbers of pounds of salt. We would then do the following:

1. Set an ammeter to zero
2. Send 2 amps of current through the ammeter
3. Send 3 more amps of current through the ammeter
4. Read the final result on the ammeter: $2 + 3 = 5$

To solve the same problem as a digital machine would, we would first have our salt in one-pound blocks. We would select 2 blocks, then 3 more blocks. Then we would observe that there were a total of 5 blocks.

To solve this problem on a digital machine, we would program the machine to add and give the machine the ***numbers*** 2 and 3. The machine would then give the answer 5.

The basic difference between the machine can be summarized as follows:

An analogue machine accepts ***quantities***
A digital machine accepts ***numbers***

BOOK III

How Computers Make Logical Decisions

PREFACE

In Books I and II the general operating characteristics of the MINIVAC 601 and large commercial computers were examined. Emphasis was placed on the nature and function of components which are combined to make up a computer system. This book assumes a basic knowledge of the functions of the components of digital computer systems such as would be obtained from reading Books I and II.

In this book attention is focused on the process and techniques by which computers are able to perform operations which we might describe as thinking. There is a great deal of question as to whether computers actually "think". Discussion of this question is deferred until later in this book after you have done some of the experiments. Most of what we speak of as "thinking" is a process of searching memory or the environment for facts and making decisions based on those facts. In this book you will learn how computers make decisions based on such facts. The concepts associated with computer logic and the techniques which computers use to solve problems and make decisions will be explored in some detail.

1. BASIC OPERATIONS

As you have undoubtedly come to realize, the modern-day computer performs practically all of its operations using the zeros and ones of a binary code. In Books I and II the reason for this use of binary coding was explained with reference to the "bi-stable" elements used for control, processing and storage in large scale computers and the MINIVAC 601.

In the experiments which you have already performed, the lights were interpreted as a 1 when they were ON and a 0 when they were OFF; the pushbuttons indicated a 0 when they were UP and a 1 when DOWN; the relays stored a 1 when they were ON and a 0 when OFF. Through your experience with the lights, pushbuttons and relays, you should be well acquainted with the on-off (0-1) nature of the digital computer.

As indicated above, this book explores the logical operations used by a digital computer and examines how computers "think" and make decisions. Logical operations, as all other computer functions, are performed using the 0's and 1's of the binary code. The nature of this "thinking" process can best be demonstrated with some simple examples.

The Operation "AND"

In beginning to examine computer thought processes, it may be helpful to use examples in order to develop some basic logical elements. These can then be used to solve more complicated and sophisticated problems.

As an example of a logical operation, consider the following series of statements:

A. If I go outdoors
and
B. If it is raining
then
C. I will get wet.

Here we have three ideas which combine in a "logical" way to produce a conclusion which, according to the rules which we use for thinking, is "true". In order to convey the basic characteristics of this combination of ideas to the computer, we must find a way to represent the basic relationships which link the three sentences in the example above using the 0-1 binary code with which the computer is able to work.

Let us look at the sentences again. There are three statements related in such a way that the third statement can be arrived at as a valid conclusion, given the information supplied by the first two sentences. To put it another way, if the first two statements are "true", then the relationships between the first two sentences and the third sentence are such that the third sentence must also be true. On the other hand, neither the first sentence nor the second sentence alone leads to a logical conclusion.

In order to make this point obvious, we might restate the three statements above in the following form:

A. If it is true that
I go outdoors
AND
B. If it is true that
it is raining
THEN
C. It will be true that
I will get wet.

If we now ignore the content of the three sentences and look only at the relationships which exist between A, B, and C we can state the general condition that

If A is true
AND
If B is true
THEN
C is true.

The relationship linking statements A and B with statement C establishes an exclusive set of conditions which will be satisfied ***only*** when statement A and statement B are ***both*** true. It follows from the above that:

If A is false
AND
If B is true
THEN
C is false

or,

If A is true
AND
If B is false
THEN
C is false.

To relate the contents of these logical statements to the binary coding system used by the computer, the following relationships may be defined.

1 = true
0 = false (not true)

Using this coding system and remembering that (a) pushing button DOWN indicates as 1, leaving it UP indicates a 0, and that (b) a light ON indicates a 1, a light OFF indicates a 0, we can program the computer so that the true or false conditions for statements A and B can be given to the computer as input and the computer will give us the condition of the third statement as true or false output:

```
1+/1Y
1×/2Y
2×/3A
3B/3—
```

COMPUTER REPRESENTATION OF THREE STATEMENTS

Using the circuit above, pushbutton 1 will be used to communicate the true or false condition of statement A and pushbutton 2 will be used to communicate the condition of statement B. If pushbutton 1 is DOWN and pushbutton 2 is UP, statement A will be true and statement B will be false. When both pushbuttons are held down, both statements A and B will be indicated as true. In summary:

	PUSHBUTTON 1	PUSHBUTTON 2	LIGHT 3
Up or Off	Statement A is false	Statement B is false	Statement C is false
Down or On	Statement A is true	Statement B is true	Statement C is true

Pushbutton—Light Equivalents

Using the circuit and representation indicated above, pushing ***both*** pushbutton 1 and pushbutton 2 indicates to the computer that both statements A ***and*** B are true—it is true that I am going outside and it is true that it is raining. The computer then indicates that, given the information which we have provided as input, it can indicate that statement C must also be true—it is true that I will get wet.

This circuit is an example of a basic relationship which will be found in computer handling of many logical problems. Certain information about the relationships between the elements (statements) in the logical problem is supplied to the computer by a program. We supplied this information about the relationship between statement A, statement B, and statement C by wiring the circuit indicated onto the console of the MINIVAC. Once the computer has this information—once the computer is "programmed"—it is ready to use the information about statements A and B to reach a conclusion about statement C. The program tells the computer that ***both*** statements A ***and*** B must be true for statement C to be true. Therefore ***both*** pushbutton 1 and pushbutton 2 must be pushed before light 3 comes ON.

This simple combination of three statements demonstrates one of the basic operations which a computer must be able to perform if it is to solve logical problems. This operation is called "AND". Statements A ***and*** B must both be true for statement C to be true.

Just as symbols are used as a shorthand to represent the ideas in arithmetic ("+" represents the idea of addition) symbols are used to represent the operations involved in solving logical problems. In the case of the concept AND the symbol X is used as a symbol:

$$X = \text{AND}$$

Thus, the relationships between statements A, B and C may be stated as:

$$A \, X \, B = C$$

Using this notation, in combination with the definitions of 1 and 0 as true and false (not true) respectively, the conditions expressing the relationships linking the three statements above can be expressed as:

$$1 \, X \, 1 = 1$$
$$0 \, X \, 1 = 0$$
$$1 \, X \, 0 = 0$$
$$0 \, X \, 0 = 0$$

The equals sign (=) in these equations may be read as "THEN" rather than as "equals". The relationships expressed by each of the four logical statements above are:

A. If I go out
AND
B. If it is raining (1 X 1 = 1)
C. I will get wet.

A. If I do not go out
AND
B. If it is raining (0 X 1 = 0)
THEN
C. I will not get wet

A. If I go out
AND
B. If it is not raining (1 X 0 = 0)
THEN
C. I will not get wet.

A. If I do not go out
AND
B. If it is not raining (0 X 0 = 0)
THEN
C. I will not get wet.

In the program for the computer representation of the three statements, the condition "I go out" is represented by pushbutton 1 and the statement "It is raining" is represented by pushbutton 2. The light represents the statement "I will get wet". Remembering that a 1 means that a pushbutton is DOWN or a light is ON and a 0 indicates that the pushbutton is in UP and that the light is OFF, we developed a computer program based on the four equations above.

The Operation "OR"

Just as arithmetic uses several different kinds of operations (addition, subtraction, division, multiplication) to solve arithmetic problems, the solution of logical problems similarly involves the use of several different logical operations. We are now familiar with one of these operations—the operation "AND" represented by the symbol X.

A second operation of basic importance is the "OR" operation. Just as the "AND" operation represents the idea of two things both being true, the "OR" operation represents the idea of ***either*** of two or more things being true.

Using an example similar to our previous three statements, we can consider the following:

A. If I stand in the rain
OR
B. If I stand in the shower
THEN
C. I will get wet

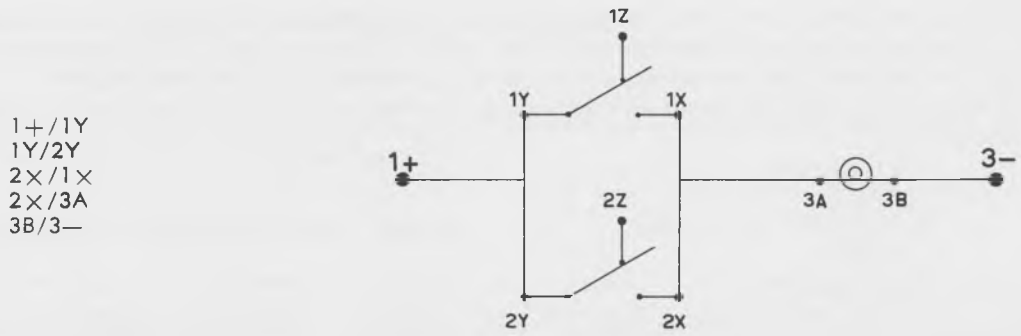
Logical representation of the relationships between these three statements may be summarized using the symbol + to mean "OR".

$+ = \text{OR}$

And the relationships between the statements may be stated as:

$A + B = C$

These relationships can be represented to the computer using the following program:



COMPUTER REPRESENTATION OF THREE "OR" STATEMENTS

This program establishes the condition of the three statements with pushbutton 1 representing statement A, pushbutton 2 representing statement B and light 3 representing statement C. When pushbutton 1 is pushed, statement A is indicated as true. When pushbutton 2 is pushed, statement B is indicated as true. When light 3 is ON, statement C is indicated as true.

"AND" and "OR"

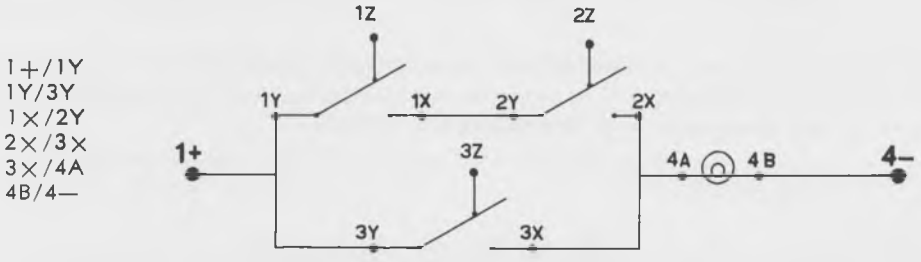
The concepts of "AND" and "OR" can, of course, be combined to express somewhat more complicated conditions for a logical problem. You may wish to experiment with some circuits representing the conditions of various combinations of "AND" and "OR" relationships. As an example of one such combination, consider a combination of the two sets of statements used separately for the "AND" and "OR" examples thus far:

A. If I go outside
   AND
B. If it is raining
   OR
C. If I stand under the shower
   THEN
D. I will get wet.

This combination of relationships may be expressed symbolically as:

A X B + C = D

The program and circuit representing this set of relationships in indicated below. Pushbutton 1 is used to represent statement A, pushbutton 2 to represent statement B, pushbutton 3 to represent statement C and light 4 to represent statement D.



COMPUTER REPRESENTATION OF "AND" AND "OR" STATEMENTS

The Operation "NOT"

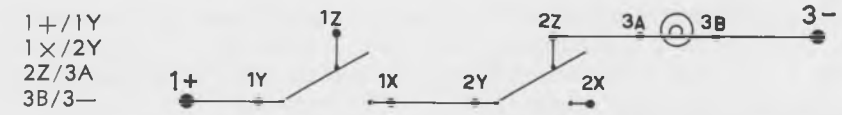
The logical operations defined thus far have each represented a concept which was basic to the development of logical reasoning. The "AND" operation represents the concept of combination. The "OR" operation represents the concept of alternative. The "NOT" operation represents an equally important, but somewhat different concept.

In the case of "AND", we were concerned with things happening together. In the case of "NOT", we are concerned with the reverse of the "AND" situation. The "NOT" operation is used to deal with the concept that something will happen ***if*** something else does ***not*** happen.

Returning to the familiar example concerning the wetness of rain, consider the following series of statements;

A. If is it raining
   AND
B. If I am NOT under cover
   THEN
C. I will get wet

The program and circuit representing this set of relationships is given below. Pushbutton 1 is used to represent statement A, pushbutton 2 to represent statement B, and light 3 to represent statement C:



COMPUTER REPRESENTATION OF THREE "NOT" STATEMENTS

Using the normally closed contacts of pushbutton 2, we are able to represent the concept "NOT". Light 3 comes ON if pushbutton is pushed and pushbutton 2 is NOT pushed.

Symbolically, the concept NOT is represented by a line over a symbol. For example,

$$\overline{Z} = 1$$

Thus, the relationship between the statements above may be stated as:

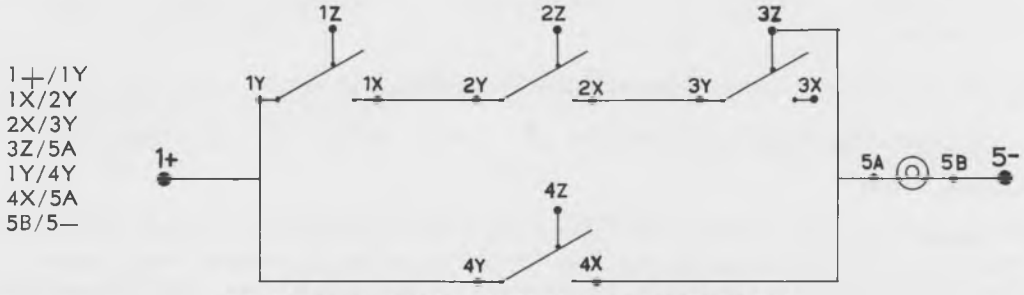
$$A \times \overline{B} = C$$

"AND", "OR" and "NOT"

The examples used to demonstrate the three operations discussed thus far can be combined further to create a program representing the following series of related statements:

A. If it is raining
   AND
B. I am outside
   AND
C. If I am ***not*** under cover
   OR
D. If I am in the shower
   THEN
E. I will get wet

In the following program, statement A is represented by pushbutton 1, statement B is represented by pushbutton 2, statement C is represented by bushbutton 3, statement D is represented by pushbutton 4, and statement E is represented by light 5.



AN "AND", "OR" AND "NOT" PROGRAM

Symbolically, the set of relationships may be expressed as:

$$A \times B \times \overline{C} + D = E$$

Note that the normally open contacts of pushbuttons 1, 2 and 4 are used, while pushbutton 3 is wired using the normally closed contacts.

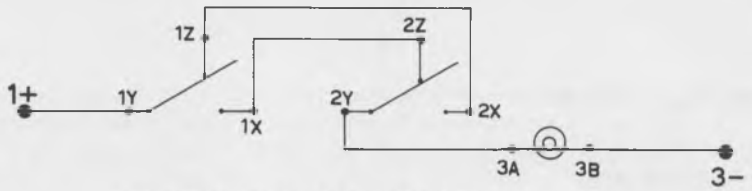
The Operation "EITHER BUT NOT BOTH"

As we begin dealing with more complicated circuits, it becomes desirable to express concepts which are somewhat more involved than the simple combinations of "AND", "OR" and "NOT" considered above. The "EITHER BUT NOT BOTH" concept provides an example of such a case. The idea to be expressed in this concept can be illustrated with reference to our ever-present wetness problem:

A. If I turn on the hot water
   OR
B. If I turn on the cold water
   THEN
C. I will have to step out of the shower.

Converting this series of statements into a program for the MINIVAC, we have the following program and circuit:

```
1+/1Y
1Z/2×
1×/2Z
2Y/3A
3B/3—
```



"EITHER BUT NOT BOTH"

Pushbutton 1 is used to represent statement A, pushbutton 2 to represent statement B, and light 3 to represent statement C. The program uses the contacts so that if ***either*** pushbutton 1 or pushbutton 2 is pushed, light 3 will come ON. Light 3 will NOT come on if ***both*** pushbuttons 1 and 2 are pushed.

Expressing these relationships symbolically requires the use of two statements, each expressing a condition which might exist:

1. Statement A, but NOT statement B, may be true:

   $$A \times \overline{B}$$

2. Statement B, but NOT statement A, may be true:

   $$\overline{A} \times B$$

Since ***either*** of the above is permissible, the entire set of relationships may be stated as:

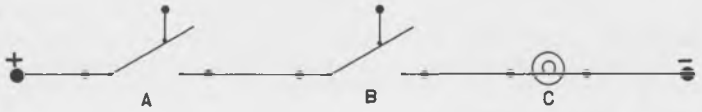
$$(A \times \overline{B}) + (\overline{A} \times B) = C$$

A Short Summary

It may be helpful to review briefly the four logical operations which have been demonstrated thus far using the pushbuttons and lights:

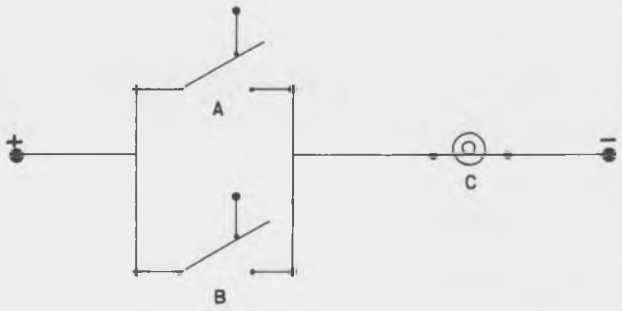
The concept ***"AND"***

$A \times B = C$



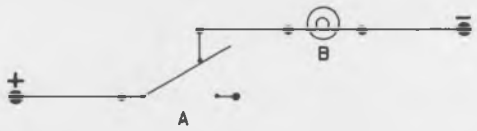
The concept ***"OR"***

$A + B = C$



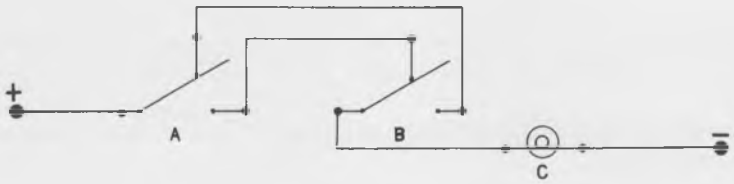
The concept ***"NOT"***

$\overline{A} = B$



The concept ***"EITHER BUT NOT BOTH"***

$(\overline{A} \times B) + (A \times \overline{B}) = C$



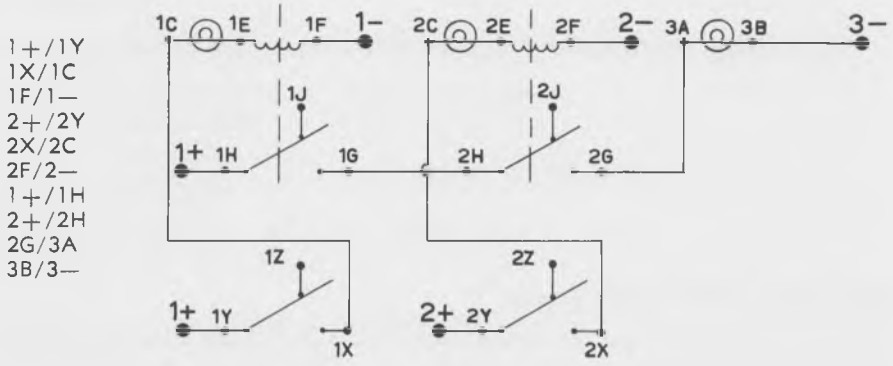
2. RELAYS FOR MORE FLEXIBLE THINKING

In Book II we discovered that the relays could be used to supply an effective binary storage system for the MINIVAC 601. At that time it was also indicated that the relays would play an important part in the processing operations of the computer. As you program more complicated problems on the MINIVAC, you will find that the relay is an extremely versatile component. In more complicated circuits, the relays will be used to perform all the basic logical functions demonstrated using the pushbuttons and lights in the earlier section. In fact, a single relay will, in some situations, be used to represent two or more logical operations at one time.

The following programs demonstrate the use of the relays to perform the basic "AND", "OR", "NOT", and "EITHER BUT NOT BOTH" functions already developed using the pushbuttons and lights.

The Relay AND Circuit

The following circuit and program represent the condition that if relay 1 is ON (=1) AND relay 2 is ON (=1) then light 3 will be ON (=1). Symbolically, A X B = C:



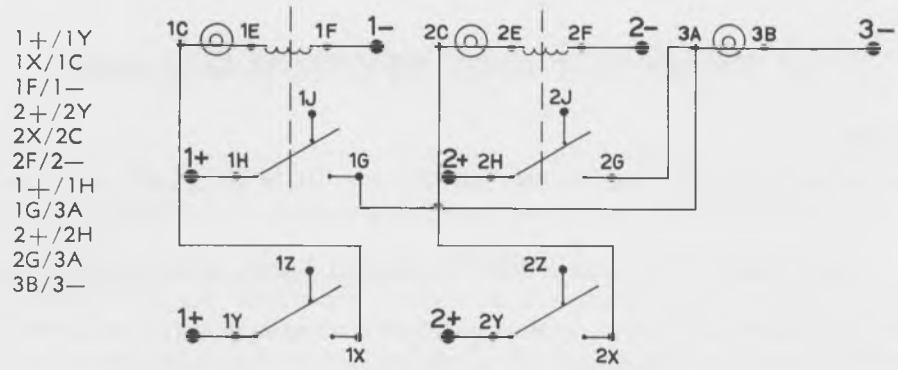
THE RELAY AND CIRCUIT

In this circuit, pushbutton 1 is used to control relay 1 and pushbutton 2 to control relay 2. Pushing pushbutton 1 causes relay 1 to close by providing current to the coil of relay 1. In a similar manner, pushbutton 2 controls the flow of current to the coil of relay 2. The normally open contacts of the two relays are used to effect the "AND" condition. Light 3 indicates when the "AND" condition is met—both relay 1 ***and*** relay 2 are closed (ON).

The Relay OR Circuit

The program and circuit below use two relays to establish the conditions for the "OR" relationship. Using this program, if ***either*** relay 1, controlled by pushbutton 1, ***or*** relay 2, controlled by pushbutton 2, are ON, then light 3 will come ON indicating that the OR condition is met. Symbolically, the condition is represented by the expression:

$$A + B = C$$

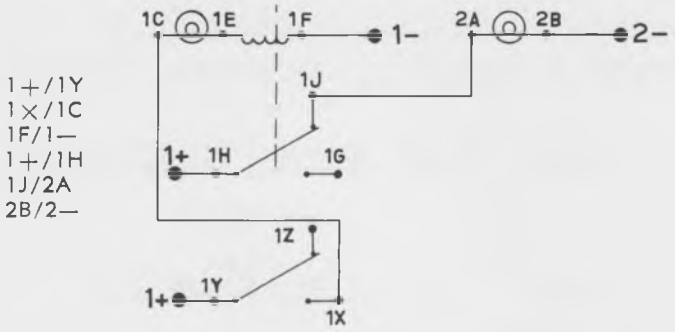


THE RELAY OR CIRCUIT

The Relay NOT Circuit

The circuit and program below represent the NOT condition expressed in the form that if relay 1 is ***not*** energized (is not ON) light 2 will come ON. Symbolically, this may be stated:

$$\overline{A} = B$$

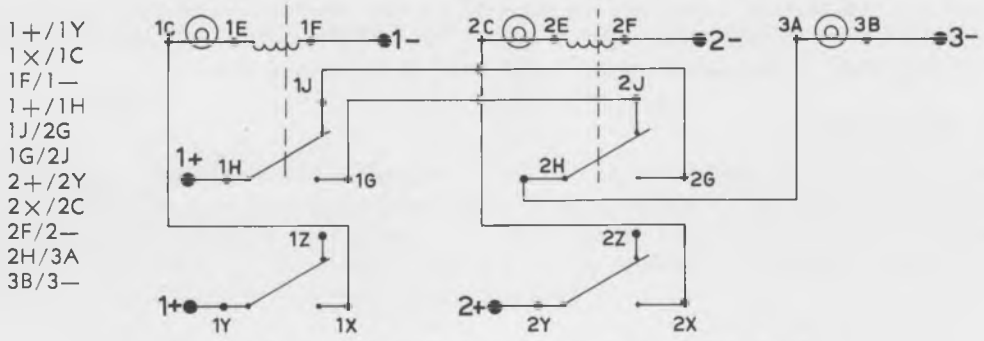


THE RELAY NOT CIRCUIT

The Relay EITHER BUT NOT BOTH Circuit

The circuit and program below represent the EITHER BUT NOT BOTH condition using relays 1 and 2 controlled by pushbuttons 1 and 2 respectively. The symbolic representation for this condition is:

$$(A \times \overline{B}) + (\overline{A} \times B) = C$$



THE RELAY EITHER BUT NOT BOTH CIRCUIT

3. THINKING AND DECISION-MAKING

Boolean Algebra

The operations with which you are now familiar are the basic operations of ***Boolean Algebra.*** Just as addition and subtraction are basic to ordinary algebra, AND and OR are basic to Boolean Algebra.

Boolean Algebra was introduced in 1847 by George Boole, an English mathematician and logician. Boole's system was developed to provide a "shorthand" for the system of logic originally set forth by Aristotle. Aristotle's system dealt with statements which were either true or false.

Only recently has Boolean Algebra become an important mathematical field. Because of its efficiency for handling any single-valued function with only two possible states, Boolean Algebra has become the basic language used to deal with the switching circuits which are basic to modern computers. The application of the operations and techniques of Boolean Algebra to switching circuits was first suggested in 1938 by Dr. Claude Shannon of the Massachusetts Institute of Technology. It was Dr. Shannon's original development work which led to the MINIVAC 601.

The value of Boolean Algebra becomes particularly apparent when complex logical problems must be organized for presentation to a computer, or when it is desirable to simplify an involved

circuit. For the purposes of this manual, the following brief introduction to Boolean algebra is sufficient:

Basic concepts

The two-state system:

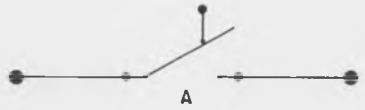
Boolean Algebra requires that any variable have only two possible states. Thus:

a statement may be TRUE or FALSE.
a circuit may be CLOSED or OPEN
a switch may be ON or OFF.

Representation and Notation:

Let: 0 = false, OFF, OPEN
1 = true, ON, CLOSED

On-off electrical circuits provide a graphic picture of the basic concepts of Boolean algebra. Consider the following circuit diagram:



Switch A is shown in the OPEN (no current flowing) position. In notational form then,

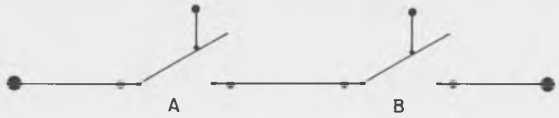
A = 0

means: switch A is OPEN. Similarly, B = 1 means "switch B is CLOSED" (that is, current is flowing through switch B).

Basic Operations:

AND:

consider two switches—A and B—connected in series:



For the ***entire system*** to be ON, current must flow through ***both*** switch A ***and*** switch B. Using the notation that:

X means AND

and Z represents the entire system then,

A X B = Z

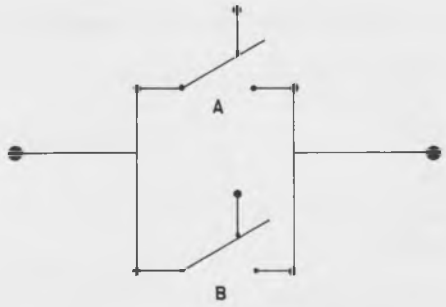
There are four possible states for the system; these possible states may be listed in a ***truth table:***

A	B	Z
0	0	0
1	0	0
0	1	0
1	1	1

(it may be helpful at this point to refer back to the original discussion of the AND circuit to see a verbal representation of the truth table).

OR:

Consider two switches (—A and B—) connected in parallel:



For the system to be ON, current may flow through ***either*** switch A ***or*** switch B. Using the notation that:

$+$ means OR

then,

$$A + B = Z$$

The truth table for this condition:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

NOT:

This operation stands for the ***opposite state*** or condition. The notation is a line over the symbol. Thus:

$\overline{A}$ means "not A"

or, in the binary system:

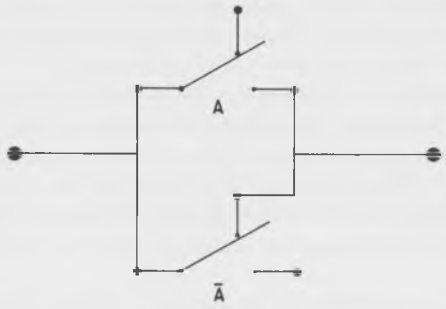
$$\overline{1} = 0$$
$$\overline{0} = 1$$

The two negation relationships are known as the ***complementarity laws:***

1) $A \times \overline{A} = 0$
$A + \overline{A} - 1$

graphically:

1) $A \times \overline{A} = 0$



2) $A + \overline{A} = 1$

Mathematical laws:

the commutative law:

1) $A + B = B + A$
2) $A \times B = B \times A$

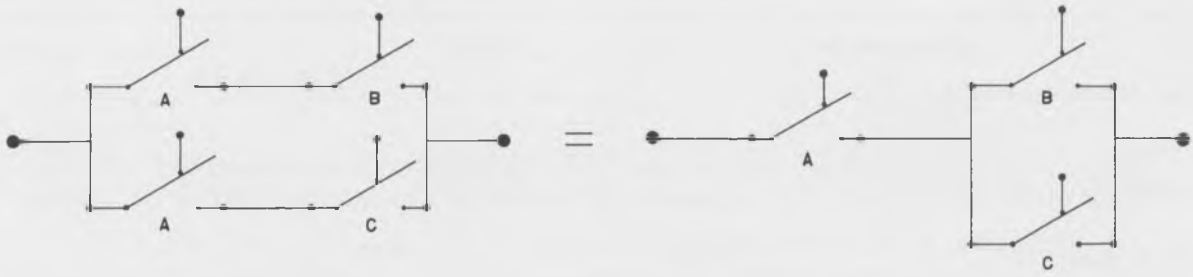
the associative law

1) $A + (B + C) = (A + B) + C$
2) $A \times (B \times C) = (A \times B) \times C$

the distributive law:

$(A \times B) + (A \times C) = A \times (B + C)$

graphically:



(A X B) + (A X C) = A X (B + C)

(The distributive law is particularly useful for circuit simplification).

Decision-Making with Insufficient Information

In the preceding section of this book, the basic elements of computer logic were discussed. Using these basic operations (the operations of Boolean Algebra) complex logical problems can be analyzed and solved in much the same way that complex mathematical problems are solved using the basic arithmetic tools—addition, subtraction, multiplication and division.

The final section of this book is devoted to a series of logical, decision-making, problems. Each problem will be considered in some detail and each step in the solution of the problem will be discussed. The basic operations developed in the preceding sections will be applied to each problem to obtain a computer program for the solution of the problem.

MINIVAC 601, just like a large digital computer, can be used to solve any logical problem if the following two requirements are met:

1) if the computer has sufficient capacity for all conditions of the problem.
2) if the computer is given sufficient information for the solution of the problem.

The first requirement—sufficient capacity—can generally be met by using a computer of a size appropriate to the problem. Theoretically, at least, any logical problem can eventually be solved on a computer if the computer is large enough.

The second requirement—sufficient information—***must*** be met if the computer is to produce a solution to a logical problem. A computer cannot guess. If all information about a particular problem is not available, assumptions about the missing information must be made and presented to the computer. The computer will then be able to solve the problem, basing its solution on the facts and assumptions it has been given. Without sufficient information—either in the form of facts or assumptions—a computer cannot reach a solution to a problem. But then, neither can a person.

For an example of a problem to be solved with insufficient information, let us return to the problem of raining and wetness. Given only the fact that it is raining outside, we cannot reach a conclusion as to whether or not we are wet; nor can a computer be programmed to determine whether or not we are wet. The information is insufficient.

However, if we ***assume*** that A) we are outside and B) we are not under cover, then we can conclude that we are wet. On the other hand, if we ***assume*** that A) we are inside or B) we are under cover, then we can conclude that we are not wet. Therefore, if we do not have sufficient information, we can make assumptions about our problem and arrive at a series of conclusions. Similarly, we can program a computer with both the information and the assumption to yield ***conditional*** answers.

Fact: It is raining outside.

Assumptions	Conclusions
I am outside AND I am not under cover	I am wet
I am inside OR I am under cover	I am not wet

When analyzing a problem for computer solution, the distinction between fact and assumption must always be kept in mind. Otherwise, the solution will be incorrect and/or incomplete.

Simulation

Complex problems for which a trial-and-error solution would be too costly or time-consuming are often ***simulated*** on a computer. When a computer simulation is used, the conditions of the problem are presented to the computer and alternative solutions are tried. When the desired simulated result is achieved, the method of solution or the actual simulated solution can be applied to the real world situation.

For example, computer simulation is used effectively to test the strength of a rocket design under the conditions of outer space—without actually building the design or sending it into space. The rocket design is programmed onto a computer as a series of complex mathematical relationships in terms of geometrical shape, resistance to friction, reaction to pressure, and many other important details. Once this information has been given to the computer, the computer is considered to be a "simulated rocket".

To test the design, the engineers and scientists communicate varying conditions to the "simulated rocket", and the computer communicates back the results of the conditions. For instance, the "simulated rocket" may be given a series of accelerations; the results of these accelerations will be a series of data which, when properly interpreted, will show what happened to the various parts of the rocket. Similarly, various materials can be tested under outer space conditions.

The result of a simulation as involved as a rocket design is a vast amount of information which must be carefully interpreted and analyzed before the final answer can be achieved.

For a simple example of some of the advantages and problems of a computer simulation, let us return to the rainstorm. And, let us now assume that we are wearing a new suit. The problem confronting us is, of course, to stay dry. We could easily enough stay inside, but that would be impractical. On the other hand, we could step outside—which would be fine ***unless*** it is raining. Obviously, we do not want to stay inside; but at the same time, we do not want to risk getting our new suit wet.

Using simulation, we can try various solutions to our problem without risking getting wet. Then we can analyze the results of the simulation and find the best plan of action.

Problem: I must not get wet
Possible Solutions:

A. I can stay inside
B. I can go outdoors
C. I can wear a raincoat
D. I can carry an umbrella

We must now find the results of each possible solution:

Possible Solutions	**Results**
A. Stay inside	I will not get wet
B. Go outdoors	I will not get wet ***if*** it is not raining ***or*** if I am under cover.
C. Wear a raincoat	I will not get wet
D. Carry an umbrella	I will not get wet

This is the form in which computer results of a simulation generally appear. It is left to the persons involved to select the best possible solution. To choose the best possible solution, both the solution and the result must be considered. For instance, these are our alternatives and with them are other factors which will influence our final decision:

Alternatives	**Results**	**Other Factors**
A. Stay inside.	I will not get wet.	I have an appointment I must keep.
B. Go outdoors.	I ***may*** not get wet.	I cannot afford a new suit.
C. Wear a raincoat.	I will not get wet.	I do not have a raincoat.
D. Carry an umbrella.	I will not get wet.	I feel silly carrying an umbrella.

At this point, it is up to each of us to decide on a best possible solution. Alternative B is not the best possible solution, nor is alternative C—each will probably cost us money. This leaves a choice between A and D. If the appointment ***must*** be kept, then the only alternative left is D. Although we will feel silly carrying an umbrella, this is the ***best possible*** solution to our problem—a solution which we have achieved without ruining a suit in the process.

Real problems are, of course, much more complex than our simple example. However, our example had the same advantage as a large-scale simulation: we tested various situations ***before*** actually trying them out, and thus saved ourselves the expense of replacing the item under test if one of our solutions was incorrect. That is, we now ***know*** that we will not get the new suit wet when we go outdoors because we will be carrying an umbrella.

Of course, we had to have the possible solutions in mind before we could try them out. This is one of the difficulties of a simulation technique: to find the best possible solution, all possible solutions must be tried. It is up to the persons involved in the problem to determine the alternatives. The computer will yield the results of each alternative, and then it is once again up to the persons involved to decide upon the best possible alternative.

Thoughts About Thinking

Before using MINIVAC 601 to solve problems which require reasoning and "thought" for solution, we might pause for a moment to think about ***thinking.***

Does a computer really "think"? If by "think" we mean does the computer follow a series of steps which are logically related and which lead the computer to the solution of a problem, our answer must be yes. In this sense, the computer definitely does "think".

If, on the other hand, we consider thinking in the sense of creative human thought in which ideas or thoughts are conceived, the computer does not "think". The computer does not "know" what it is thinking. It has no consciousness. A computer must (at least initially) be programmed to perform each step in the solution of a problem. It can do ***only*** what the computer operator or programmer tells it to do. Although we may not know the solution to a problem, we can decide on the steps which should be taken to reach the solution and can communicate these steps to the computer.

The reason that the computer can solve the problem which we may not be able to is that the computer remembers each step in detail and does not become "confused". While we are talking about the fourth step in a problem, we may have forgotten some detail of the first step which is necessary in order to arrive at the correct solution. The computer will not forget.

The point of this comment is that the computer must be programmed with absolute accuracy. The computer does not know what it is to do; it has no way of judging when it has made a mistake. It will simply follow the instructions it is given (by programming) and arrive at the conclusion which is the result of following these steps.

You may have noted the comment that the computer must, ***at least initially,*** be told each operation which it is to do. The word "initially" must be used because a computer can be programmed to program itself. That is, a computer can be given instructions for following a series of steps or operations which will enable it to generate more instructions which it can then perform. A computer, once it has been given such a series of instructions and has been provided with additional instructions which enable it to "evaluate" the conclusions which it reaches —to determine whether the solutions reached meet previously supplied criteria—can begin to "learn" concepts and relationships which the person originally programming it did not know or, at least, did not communicate to it.

In many ways, the process by which a computer is educated—or taught to think— is similar to that used to teach a child. The computer and child must both be taught to gain information from their environment, to interpret it, and to use the information in association with data drawn from past experience to arrive at conclusions about a given situation or the solution to a particular problem.

As you consider the programs which must be given to MINIVAC 601 to enable it to solve the problems and riddles in the following section, consider the information you would have to give a person who had never before solved a problem to enable him to duplicate the performance of the computer.

As human beings, we can do many things; but we sometimes forget that many people and much experience have gone into the teaching process through which we have received and stored in our memories the data and instructions which enable us to deal with our environment.

4. SOME COMPUTER PROBLEMS

A Mind Reading Program

As an initial example of the use of logical operation to reach a decision, consider the following situation:

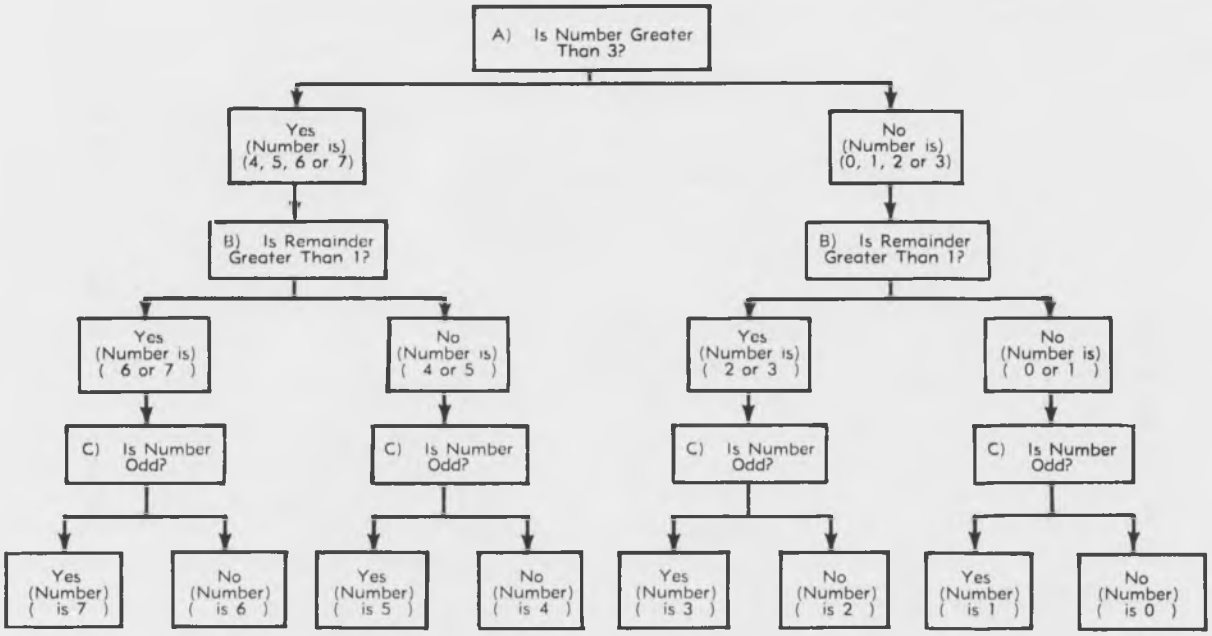
If a person thinks of a number between 0 and 7, is it possible, by asking three "yes-no" questions, to determine the number of which he is thinking?

This problem can, in fact, be solved without resorting to mind reading. Through a careful choice of questions we can eliminate all but the correct answer through a series of successive logical operations. The three questions which provide the key to the problem are:

A. Is the number greater than 3?
B. When the number is divided by 4, is the ***remainder*** greater than 1? (For example, if 6 is divided by 4 the answer is 1 plus a remainder of 2; similarly, 1 divided by 4 gives an answer of 0 plus a remainder of 1.)

C. Is the number odd?

The process followed in solving this problem can be analyzed using a ***flow chart.*** This chart indicates each step in the decision process and shows the sequence which leads to the final result. A flow chart for the solution of this problem appears below:



The flow chart shows that question A—is the number greater than 3—separates the possible numbers into two groups: those greater than 3 (4, 5, 6, 7) and those not greater than 3 (0, 1, 2, 3). Question B then separates each of these groups into two groups, and question C determines which number in the appropriate set is not excluded—in other words, determines the number of which the person was thinking.

The conditions for each of the numbers can be expressed as logical operations as follows: Let:

A = the number is greater than 3
$\overline{A}$ = the number is ***not*** greater than 3
B = the remainder is greater than 1
$\overline{B}$ = the remainder is ***not*** greater than 1
C = the number is odd
$\overline{C}$ = the number is ***not*** odd.

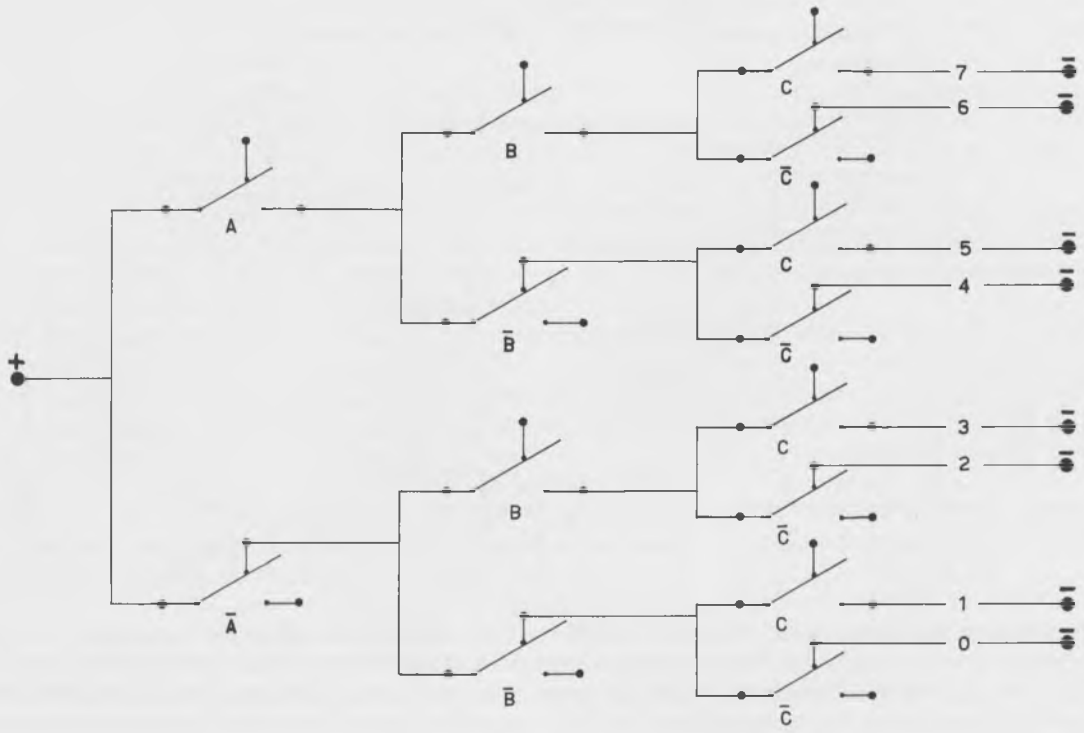
Using these definitions, we can express the conditions for each of the possible numbers as follows:

$A \times B \times C = 7$
$A \times B \times \overline{C} = 6$
$A \times \overline{B} \times C = 5$
$A \times \overline{B} \times \overline{C} = 4$
$\overline{A} \times B \times C = 3$
$\overline{A} \times B \times \overline{C} = 2$
$\overline{A} \times \overline{B} \times C = 1$
$\overline{A} \times \overline{B} \times \overline{C} = 0$

To program the problem for solution on a computer, each relationship is connected into a circuit representing the equation. Then these circuit representations of the equations must be combined for a complete program.

The circuits for the individual equations are simply combinations of the AND and NOT circuits. For example, the circuit representation for an answer of 5 would be:

Combining the eight equations above into a single circuit will give the following:



Notice that the circuit has the same fan-shaped appearance as did the flow chart. In essence, this circuit diagram is simply a circuit representation of the flow chart. Each "gate" (switch) represents an answer to one of the questions.

However, if we examine this circuit representation, we see that there are four contacts to the gate (switch) representing statement C, and there are two contacts to the gate representing statement B. If we were to program this circuit onto a computer then, we would have to have one gate which was capable of accepting four separate inputs. This can be done on the MINIVAC, and the circuit could be programmed directly from the above circuit representation.

It is possible, though, that we would want to program this problem onto a computer which had only three input contacts for each gate. What would we do in this case? Or similarly, what would we do if we wanted to keep a contact free for additions to the circuit?

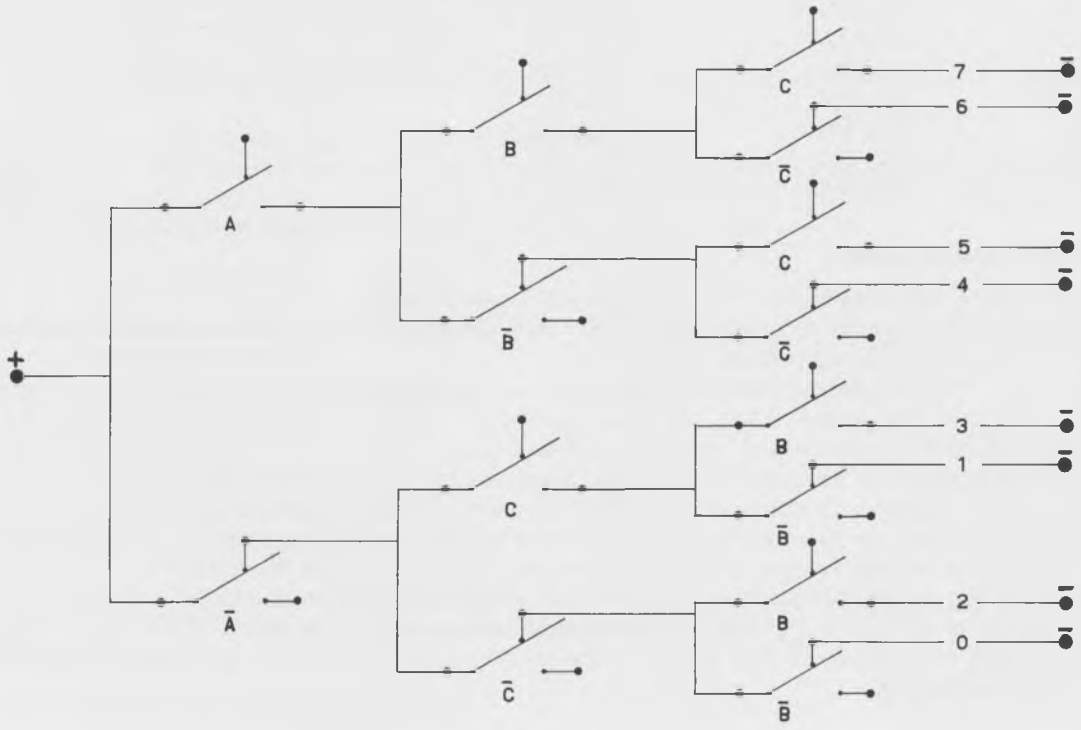
The commutative law of Boolean Algebra referred to in the previous section can be applied to this particular problem. The commutative law states that:

A X B is the same as B X A

So, we can restate four of our equations as follows:

$\bar{A} \times B \times C = 3$	becomes	$\bar{A} \times C \times B = 3$
$\bar{A} \times B \times \bar{C} = 2$	becomes	$\bar{A} \times \bar{C} \times B = 2$
$\bar{A} \times \bar{B} \times C = 1$	becomes	$\bar{A} \times C \times \bar{B} = 1$
$\bar{A} \times \bar{B} \times \bar{C} = 0$	becomes	$\bar{A} \times \bar{C} \times \bar{B} = 0$

The circuit representation for the eight equations now becomes:



This is a simple example of the manipulation of circuits which professional programmers do to fit a logical solution to the capabilities of a particular computer. Some complex logical problems can be programmed for computer solution only by making maximum use of the computer's capacity in this fashion.

We shall use a program based on this latter circuit representation for solving the problem on MINIVAC 601. The slide switches will be used for data input and data storage; the relays will be used for processing and operating storage; the rotary switch will be used for output. A pushbutton will be used for the instruction: move final answer from operating storage (relays) to output (rotary switch).

In summary then, the rules for using the "Mind Reading Program" are:

Let:

Slide Switch 4 represent the answer to question A
Slide Switch 5 represent the answer to question B
Slide Switch 6 represent the answer to question C.

A slide switch LEFT indicates a YES answer; a slide switch RIGHT indicates a NO answer. The questions are:

A (slide switch 4): Is the number greater than 3?
B (slide switch 5): When the number is divided by 4, is the remainder greater than 1?
C (slide switch 6): Is the number odd?

After the answer to each question has been communicated to the computer through the slide switches, pushing pushbutton 6 will instruct the computer to indicate the final answer on the decimal input-output dial.

A program for the solution of this problem follows:

5V/4S	5U/5C	6T/5L	D6/5K
4R/4A	5F/6F	M+/6Y	D7/5G
4B/4—	6F/6—	6X/D17	
5C/5A	M—/D18	D16/D19	
5B/5—	D18/4V	D0/6N	
6C/6A	4U/6S	D1/6K	
6B/6—	4W/5S	D2/6J	
6X/6V	5R/6H	D3/6G	
6V/5V	5T/6L	D4/5N	
6U/6C	6R/5H	D5/5J	

To use the program:

1. Ask a friend to think of any number between 0 and 7.
2. Ask this person to answer "yes" or "no" to the following questions about the number:
   A. Is the number greater than 3?
   B. When the number is divided by 4, is the ***remainder*** greater than 1?
   C. Is the number odd?
3. Indicate the answers to the questions as follows:
   If the answer to question A is YES, move slide switch 4 to the LEFT.
   If the answer to question A is NO, move slide switch 4 to the RIGHT.
   If the answer to question B is YES, move slide switch 5 to the LEFT.
   If the answer to question B is NO, move slide switch 5 to the RIGHT.
   If the answer to question C is YES, move slide switch 6 to the LEFT.
   If the answer to question C is NO, move slide switch 6 to the RIGHT.
4. Push pushbutton 6. The pointer knob of the rotary switch will turn to the number your friend was thinking of.

Book VI—MINIVAC GAMES—includes a variation on this "Mind Reading Program" which uses names in place of numbers.

Quantity Recognition

As a further example of the use of logical operations, let us consider how a computer could be programmed to recognize quantities. The problem will be to instruct the computer to determine the number of slide switches which are ON (in the LEFT position) regardless of which slide switches are used.

Quantity recognition circuits have many applications in computer logic. They can be used, for instance, to perform additions, to indicate the results of a game or a vote (win, lose, draw) or to give direct count for a quality control system. Quantity recognition circuits of this type are based on what mathematicians call a "symmetric function lattice". For our purposes, we may consider the circuit as a series of "yes-no" decisions.

Slide switches 2 through 6 will be used to provide input: a slide switch LEFT is ON, a slide switch RIGHT is OFF. Lights 1 through 6 will be used for output according to the following convention:

Light 1 ON indicates that 1 switch is ON
Light 2 ON indicates that 2 switches are ON
Light 3 ON indicates that 3 switches are ON
Light 4 ON indicates that 4 switches are ON
Light 5 ON indicates that 5 switches are ON
Light 6 ON indicates that No switches are ON.

To program the computer for quantity recognition, we will first describe the possible conditions as a series of logical expressions:

Let:

A = the condition that slide switch 2 is ON
B = the condition that slide switch 3 is ON
C = the condition that slide switch 4 is ON
D = the condition that slide switch 5 is ON
E = the condition that slide switch 6 is ON

Then, $\overline{A}$ = the condition that slide switch 2 is OFF, and so on.

Each of the various combinations of switches which yield a particular answer must now be stated and then converted to a circuit and a program. For example, the case in which light 6 comes ON—indicating that ***no*** switches are on—can only be expressed as:

$\overline{A} \times \overline{B} \times \overline{C} \times \overline{D} \times \overline{E}$ = light 6 ON

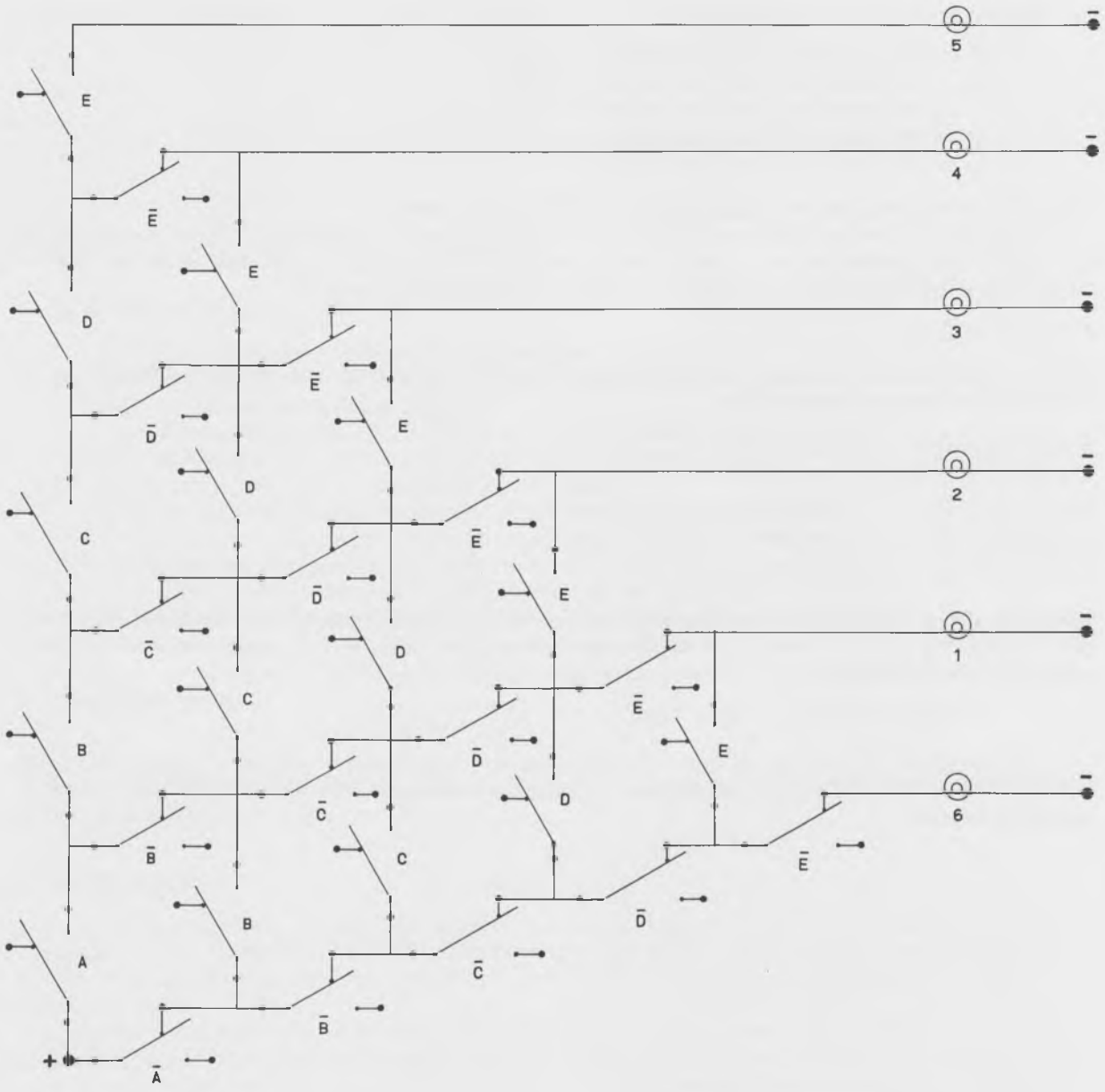
However, the possibilities for light 1 coming ON—indicating that ***one*** switch is ON—must be stated as five separate expressions:

$A \times \overline{B} \times \overline{C} \times \overline{D} \times \overline{E}$ = light 1 ON
$\overline{A} \times B \times \overline{C} \times \overline{D} \times \overline{E}$ = light 1 ON
$\overline{A} \times \overline{B} \times C \times \overline{D} \times \overline{E}$ = light 1 ON
$\overline{A} \times \overline{B} \times \overline{C} \times D \times \overline{E}$ = light 1 ON
$\overline{A} \times \overline{B} \times \overline{C} \times \overline{D} \times E$ = light 1 ON

Similarly, there will be ten separate expressions for the possible conditions for light 2 to come ON—indicating that two switches are ON, ten expressions for light 3, five expressions for light 4 and the single expression

$A \times B \times C \times D \times E$ = light 5 ON

The program and circuit for the complete solution must fit ***all*** of the conditions of the problem as stated in the 32 logical expressions of all possible cases. The full circuit may be represented as follows:



You will notice that this representational circuit has a ***lattice*** shape. Moreover, it is electrically ***symmetrical*** about the lower-left to upper-right diagonal. Hence the mathematical phrase "symmetric function lattice" referred to earlier. This circuit representation shows the conditions A and $\overline{A}$, etc., as separate switches. In the actual program, of course, ***both*** conditions for a given switch will be wired onto a single switch using the normally-open and normally-closed contacts for the A and $\overline{A}$ etc., conditions respectively.

An examination of the circuit representation shows that for conditions C, D, and E (corresponding to slide switches 4, 5 and 6) more contacts are required than are available on MINIVAC 601. Howover, by using the relays and controlling them by the slide switches, we will be able to meet the requirements of the problem.

The final program for the solution of the quantity recognition problem appears below. This program is based on the circuit representation above and uses the relays to supply the required additional contacts.

Program for Quantity Recognition:

1B/1—	6F/6—	5A/2G	5R/6H	5K/4N
2B/2—	6+/6S	4A/2K	5W/6L	5G/4J
3B/3—	6S/4V	3A/2N	4T/6K	4N/3H
4B/4—	4V/3V	2A/2T	4R/6J	4J/2L
5B/5—	3V/2V	1A/3J	6N/5L	4G/2H
6B/6—	2U/2C	6A/3N	6J/5H	4K/2S
2F/3F	2C/3C	6T/5V	6G/4H	3J/3K
3F/4F	3U/4C	6R/5S	6K/4L	3G/2T
4F/5F	4C/5C	5U/5T	5J/4K	2J/2K
5F/6F	4U/6C	5U/4S	5N/3L	2N/2R

To use this program:

1. Set slide switches 2 through 6 to the RIGHT (the slide switches are now OFF).
2. Turn power ON. Light 6 will come ON indicating that NO switches are ON.
3. Set ***any*** two of the slide switches 2 through 6 to the LEFT. Light 2 will come ON indicating that two switches are ON.
4. Set all switches to the RIGHT again. Once again, light 6 will come ON indicating that NO switches are ON.
5. Select various combinations of switches and note that in each case the output lights indicate how many switches are ON (to the LEFT). There are 120 possible combinations of switches, and in each case MINIVAC 601 will recognize how many switches are ON, regardless of which switches are used.

A Problem Involving Three Girls

Logical riddles provide apt illustrations of problems which can be programmed for computer solution. For many riddles, there are two basically different methods for handling the computer solution: the computer can be programmed with the conditions of the problem so that various solutions may be tried and the computer will indicate whether or not the attempted solutions are correct; or, the computer can be programmed to give the final solution to the problem immediately.

As an example of a logical riddle, consider the following information about three girls named Camille, Sara and Wanda:

> If Sara shouldn't, then Wanda would. It is impossible that the statements "Sara should" and "Camille could not" can both be true. If Wanda would, then Sara should and Camille could. Given this information, the problem is:
>
> Is the conclusion "Camille Could" valid? More simply, we are asked, "Can Camille"?

As we have mentioned, there are two methods of handling this problem. We can now follow either one of the two possible methods to solve the problem.

1. We can represent the conditions of the problem as logical expressions, convert these expressions into the appropriate circuit and program, and program the computer with the conditions so that by trying various solutions the computer will indicate to us whether or not our solutions are correct. In this case, truth or error are expressed in Boolean algebra statements which are directly transferred to the computer's contacts. The computer will then test each of our attempted solutions against the logical conditions and indicate whether or not the solution is correct. Because of the direct Boolean nature of this method, we shall refer to this method as the ***"Boolean"*** method of solution.
2. We can represent the conditions of the problem as logical expressions, convert these expressions into the appropriate circuit and program, and program the computer with the conditions so that the computer will automatically give us the final solution to the problem. In this case, the logical expressions are transferred to contact networks which will force the relays to comply with the conditions of the problem. When this program is turned on, the computer will automatically seek the correct answer. Because of the problem conditions which have been programmed onto

the computer, only the correct result will be permitted and this result will be communicated to the operator. We shall refer to this method as the ***"automatic"*** method of solution.

The basic difference between the two methods can be compared with the difference between a thermometer and a thermostat. A thermometer (the analogy of the "Boolean" solution) will determine the temperature and indicate what it is. A thermostat (the analogy of the "automatic" solution) will not only determine the temperature, but will also attempt to "correct" the temperature to match the temperature which has been "programmed" into it. For instance, if the temperature were 65°F and we had set a thermostat for 68°, the following would occur for a thermometer and a thermostat:

1. The thermometer would indicate that the temperature was 65°F. It would be up to us to do something about the fact that it was 3° cooler than what we wanted.
2. The thermostat would determine that the temperature was 65°F and would automatically turn up the furnace until the temperature reached 68°F.

Essentially then, the "Boolean" method of solution ***indicates*** truth or error; the "automatic" method of solution ***determines*** the correct solution and indicates a final result.

Returning now to the problem of the three girls, we will first represent the problem conditions as logical expressions, since both methods require this step. We will then develop programs for solution by each of the two methods.

Logical Expressions for the Problem Conditions

From the conditions of the problem, we can see that there are only two possible situations for each girl:

Camille could or could not
Sara should or should not
Wanda would or would not

Whatever other conditions may exist, we can be sure that there are only two possible conditions for each girl.

We can therefore represent the possible conditions for each girl by the following convention:

C = Camille could
$\overline{C}$ = Camille could not
S = Sara should
$\overline{S}$ = Sara should not
W = Wanda would
$\overline{W}$ = Wanda would not

There are two ways to set up the logical expressions for each problem condition. We can set up either a "truth statement" or an "error statement". A "truth statement" describes the permissible conditions; an "error statement" describes the conditions which are not permissible. The two statements are, of course, complementary.

The problem conditions can be stated as:

If Sara shouldn't, then Wanda would:

$\overline{S} \times \overline{W}$ = error or $S + W$ = truth

It is impossible that the statements "Sara should" and "Camille could not" can both be true:

$\overline{C} \times \overline{S}$ = error or $C + \overline{S}$ = truth

If Wanda would ,then Sara should and Camille could:

$W \times (\overline{S} + \overline{C})$ = error or $\overline{W} + (C \times S)$ = truth

"Boolean solution"

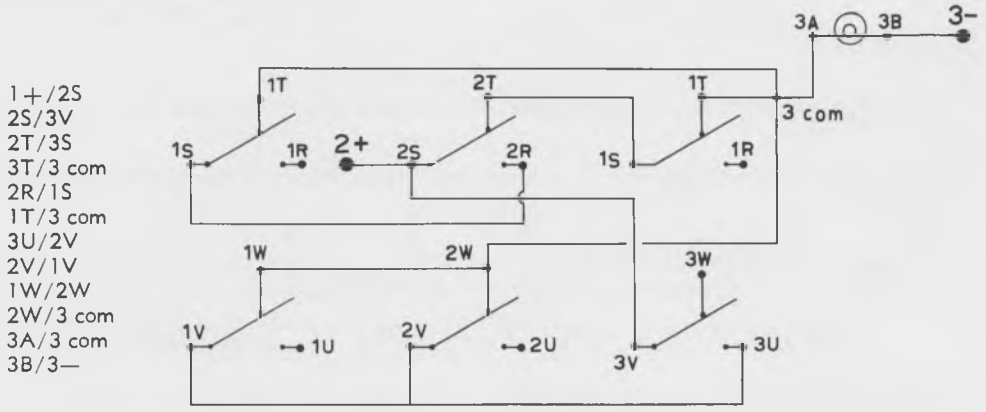
Because it is simpler in this case, we will set up a program for the "Boolean" solution from the three ***"error"*** expressions above. Slide switches 1, 2 and 3 will be used to provide the contacts for Camilla, Sara and Wanda respectively. Light 3 will be used as the error indicator. Since there are three error conditions and only two contacts for light 3, we will use the "Common" terminals to provide the necessary extra contacts. (The notation for a connection to a common terminal is "1 com", "2 com," etc.)

The circuit diagram and program for the "Boolean" solution of the problem appear below. Each of the logical expressions for the problem conditions can easily be traced on the circuit diagram.

Slide switch 1 represents Camille
Slide switch 2 represents Sara
Slide switch 3 represents Wanda

A slide switch LEFT indicates the positive condition: Camille could, Sara should, Wanda would. A slide switch RIGHT indicates the negative condition: Camille could not, Sara should not, Wanda would not.

Light 3 ON represents an error.



PROGRAM AND CIRCUIT—BOOLEAN SOLUTION

To use the program for the "Boolean" solution:

Turn power ON. Using slide switches 1, 2 and 3, try various combinations of "Sara should", "Camille could not", etc. until the error light (light 3) goes OFF. If the power is ON and light 3 is OFF, slide switches 1, 2 and 3 are indicating a correct solution to the problem. (Note: for some logical riddles there is more than one complete solution)

The problem to be solved is "Can Camille?" To discover whether or not "Camille can", the conditions of the other girls must also be considered.

"Automatic" Solution:

The program and circuit diagram for the "automatic" solution are based on the ***truth*** expressions stated above. The relays are programmed so that they are forced to follow all of the problem conditions to yield the final result. The relay indicator lights will be used to indicate the solution:

Relay light 1 will indicate the condition of Camille
Relay light 2 will indicate the condition of Sara
Relay light 3 will indicate the conditon of Wanda

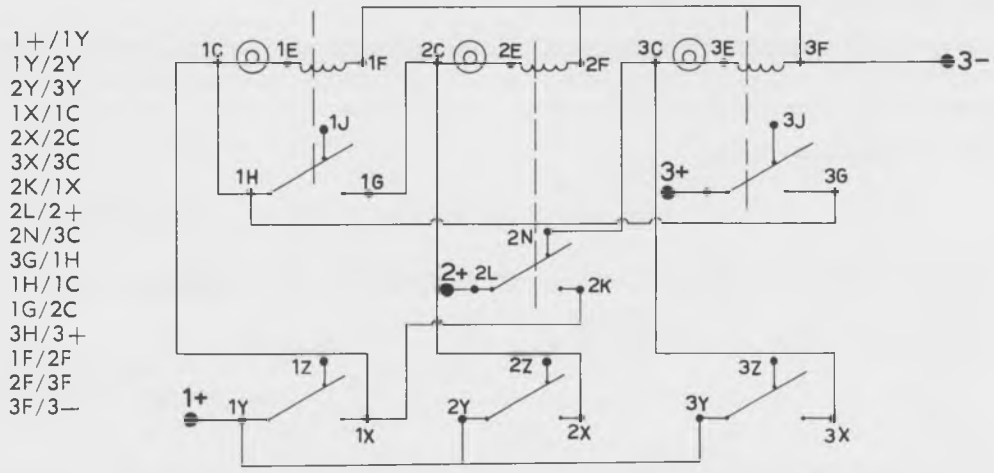
A relay light ON will indicate the positive condition; a relay light OFF will indicate the negative condition.

Since there is a possibility of more than one correct solution, pushbuttons 1, 2 and 3 are wired in so that additional information may be entered.

Pushbutton 1 represents Camille
Pushbutton 2 represents Sara
Pushbutton 3 represents Wanda

A pushbutton UP transmits no additional information to the computer. A pushbutton DOWN, however, supplies the positive condition (Camille could, Sara should, Wanda would).

When the power is turned ON, a correct solution will immediately appear in the relay indicator lights. By pushing a pushbutton, we may enter additional positive information about a girl. The relay indicator lights will then indicate the correct solution to the problem with the added information.



PROGRAM AND CIRCUIT—AUTOMATIC SOLUTION

To use the program for the "automatic solution:

Turn power ON. The solution to the problem appears in the relay indicator lights. A relay indicator light ON represents the positive condition: Camille could, Sara should, Wanda would. A relay indicator light OFF represents the negative condition.

To add positive information about a girl, push the appropriate pushbutton. The relay indicator lights will indicate the new solution, given this additional information.

The Farmer, the Goose, the Corn and the Wolf

At one time or another you have probably come across the problem of the farmer with a row boat who has to cross a river in order to reach his home on the opposite shore. The farmer has with him a goose, some corn, and a wolf. The farmer wants to bring both animals and the corn home with him, but the boat is small and will hold only one of the three objects, in addition to the farmer, at any one time. The real problem facing the farmer is that if he leaves the goose and the wolf alone, the wolf will eat the goose. And, if he leaves the corn and the goose alone, the goose will eat the corn.

The farmer's problem provides an excellent example of the advantages of simulation solution. The farmer wants to get across the river with ***all three*** of his possessions, and solution of his problem by trial-and-error in the real world could be expensive for him. Should he make a mistake, he could lose both the goose and the corn.

However, if the farmer has access to a computer, he can ***simulate*** his problem on the computer and try out various solutions until he reaches the right one. In this way, the farmer can use a trial-and-error method without any danger of actually losing either his goose or his corn.

To solve the farmer's problem on the MINIVAC, we will state the problem as a series of logical expressions. These relationships will then be converted to circuit representations and the conditions of the problem will be programmed onto the computer for solution.

The Basic Conditions:

The farmer's problem can be stated as three basic conditions:

A. The goose and the wolf cannot be left alone without the farmer.
B. The goose and the corn cannot be left alone without the farmer.
C. The boat will hold only one object in addition to the farmer.

The first two conditions—A and B— can be expressed as logical statements as follows:
Let:

F represent the presence of the farmer
$\overline{F}$ represent the absence of the farmer
G represent the presence of the goose
$\overline{G}$ represent the absence of the goose
W represent the presence of the wolf
$\overline{W}$ represent the absence of the wolf
C represent the presence of the corn
$\overline{C}$ represent the absence of the corn
E represent an error (loss of one of the farmer's possessions)

Then:

Condition A (the goose and the wolf cannot be left alone) can be expressed as:

$$G \times W \times \overline{F} + \overline{G} \times \overline{W} \times F = E$$

Condition B (the goose and the corn cannot be left alone) can be expressed as:

$$G \times C \times \overline{F} + \overline{G} \times \overline{C} \times F = E$$

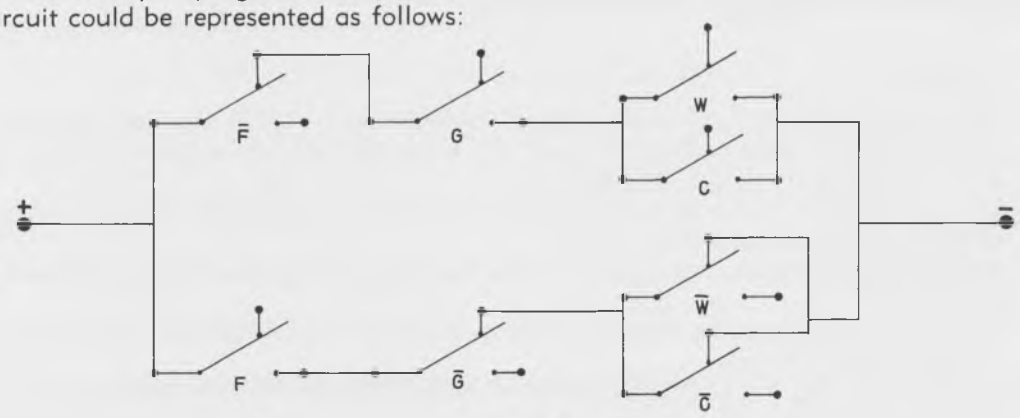
Condition C (the size limitation of the boat) need not be stated as a logical expression. The final program will include appropriate circuitry so that if the boat is overloaded, only ***one*** object will be recognized as being in the boat with the farmer.

Programming the Simulation:

The two problem conditions—A and B—can be combined as follows: $\overline{F} \times G \times (W + C) + F \times \overline{G} \times (\overline{W} + \overline{C}) = E$

This is the error condition which must be programmed onto the computer. We will program the computer so that if either error condition is met, an "alarm" light (or an "error" light) will come on.

If we had only to program the error condition, the program would be a simple one, and this basic circuit could be represented as follows:



$\overline{F} \times G \times (W + C) + F \times \overline{G} \times (\overline{W} + \overline{C}) = E$

CIRCUIT REPRESENTATION

However, this circuit represents ***only*** the processing and output portions of this program. We must also provide means of input and storage for the various solutions which we will attempt. By wiring in the slide switches and pushbuttons for input and storage, and using the relays for processing and the lights for output, we will be able to program the entire problem for computer solution.

We shall use the following conventions in the program:

An output light ON represents an object ***at home***.
A relay indicator light ON represents an object ***across*** the river.
A slide switch LEFT represents an object ***in the boat.***
A slide switch RIGHT represents an object ***not*** in the boat.
Light 5 ON represents the condition: "Wolf eats goose".
Light 6 ON represents the condition: "Goose eats corn".
Pushbutton 6 represents moving the boat ***towards home.***
Pushbutton 1 represents moving the boat ***from home.***

The following components will be used to represent the objects in the problem:

	Output Light	Relay	Slide Switch
Farmer	1	1	1
Goose	2	2	2
Wolf	3	3	3
Corn	4	4	4

Program for the solution of the Farmer's problem:

1A/1J	1U/2V	2N/3N	3N/4N	5B/5—
1B/1—	1V/1X	2W/3V	3W/4V	6B/6—
1C/1G	1X/6X	3A/3J	4A/4J	6Y/6—
1E/1U	1Y/1+	3B/3—	4B/4—	
1F/2F	2A/2J	3C/3G	4C/4G	
1H/2H	2B/2—	3E/3U	4E/4U	
1K/2K	2E/2U	3F/4F	4F/4—	
1K/1+	2C/2G	3H/4H	4H/4+	
1L/2L	2F/3F	3K/4K	4L/6A	
1N/3K	2H/3H	3L/5A		

To use the program:

1. Turn power ON. Manually push relays 1 through 4 to the LEFT. The relay indicator lights will come ON to indicate that all objects are ***across the river***. Move slide switches 1 through 4 to the RIGHT.
2. Place the farmer in the boat by moving slide switch 1 to the LEFT. (The farmer must be in the boat or the boat will not move)
3. Choose another object and place it in the boat by pushing the appropriate slide switch to the LEFT.
4. Move the boat across the river by pushing pushbutton 6. If lights 5 and 6 remain OFF, no error has been made.
5. Continue moving the boat back and forth across the river with the different objects until all four objects are at home (output lights 1 through 4 are ON.) Do not forget to remove objects from the boat when they have reached the other side (by moving the slide switch to the RIGHT)

6. If an error is made, reset the computer by setting slide switches 1 through 4 to the RIGHT and manually setting relays 1 through 4 to the LEFT.

The Television Problem

As a final example of a problem which can be programmed for computer solution, consider the following:

Barry, his wife Clara, and their children—David, Edward and Francis—are at home. It is 8:00 in the evening.

1. If Barry is watching television, so is his wife.
2. Either Edward or Francis, or both, are watching television.
3. Either Clara or David, but not both, is watching television.
4. Edward and David are either both watching or both not watching television.
5. If Francis is watching television, then Barry and Edward are also watching television.

Who is watching television?

This problem provides an excellent example of a problem which is complicated enough to be confusing to a person attempting to solve it. The problem conditions and the details of the relationships are difficult to remember. Because of its confusion-proof memory, the computer can easily handle a problem of this type.

To solve this problem, we will state the problem conditions as logical expressions and develop the appropriate circuit for a "Boolean" solution using a "truth" light to tell us when we have arrived at the correct solution. This will, of course, be a simulation technique since we will not have to actually drop in on the family to find out who is watching television.

Logical expressions for the Problem Conditions:
Let:

B = the condition that Barry is watching television
C = the condition that Clara is watching television
D = the condition that David is watching television
E = the condition that Edward is watching television
F = the condition that Francis is watching television

Then:

$\overline{B}$ = the condition that Barry is NOT watching television, etc.

And: T = truth (permissible condition)

1. If Barry is watching television, so is his wife:

   $\overline{B} + C = T$

2. Either Edward or Francis, or both, are watching television:

   $E + F = T$

3. Either Clara or David, but not both is watching television:

   $C \times \overline{D} + \overline{C} \times D = T$

4. Edward and David are either both watching or both not watching television:

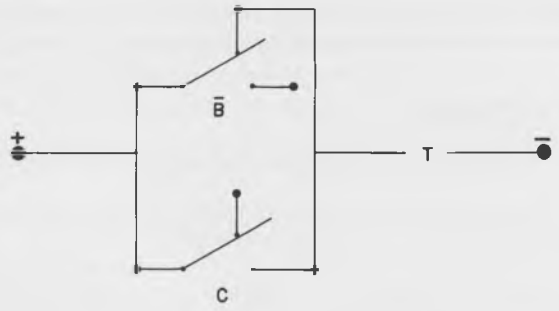
   $E \times D + \overline{E} \times \overline{D} = T$

5. If Francis is watching television, then Barry and Edward are also watching television:

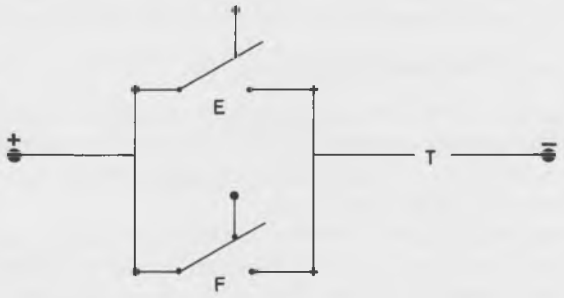
   $\overline{F} + (B \times E) = T$

Each of these expressions can be represented as logical circuits in the following manner:

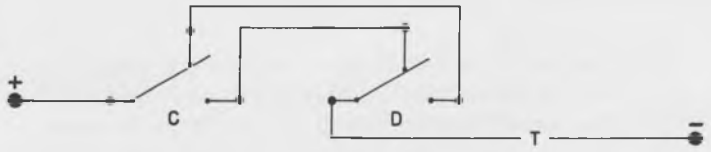
1. $\overline{B} + C = T$



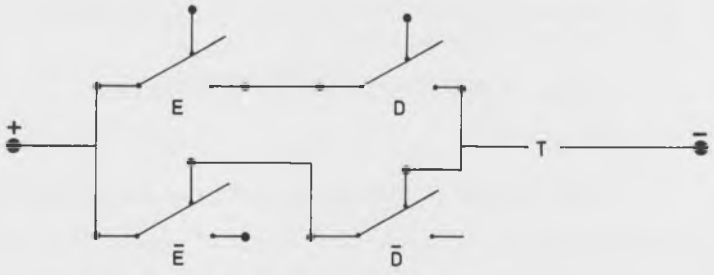
2. $E + F = T$



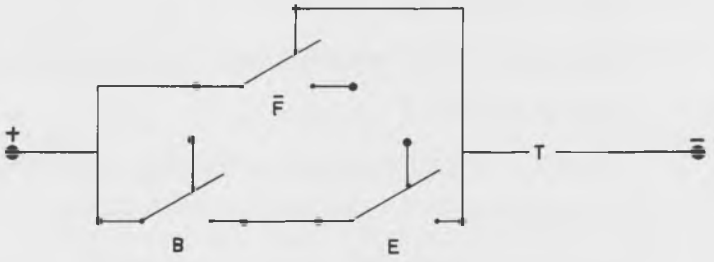
3. $C \times \overline{D} + \overline{C} \times D = T$



4. $E \times D + \overline{E} \times \overline{D} = T$



5. $\overline{F} + (B \times E) = T$



The Program for a "Boolean" Solution:

To convert the representational circuits to a program, we will use the following convention: Let:

Slide switch 1 represent Barry
Slide switch 2 represent Clara
Slide switch 3 represent David
Slide switch 4 represent Edward
Slide switch 5 represent Francis

A slide switch LEFT indicates that the person is watching television; a slide switch RIGHT indicates that the person is NOT watching television.
Light 4 will indicate the correct solution: light 4 ON indicates that the solution represented by the slide switches is CORRECT; light 4 OFF indicates that the solution is NOT CORRECT.
The complete program for the "Boolean" Solution is:

1U/1S	2W/3U	4B/4—
1S/2S	3R/4R	4U/5V
1T/2R	3S/3V	4U/1V
2R/2V	3T/4T	4V/4+
2U/3W	4A/4S	5W/1U

To use the program:

Set slide switches 1 through 5 to the LEFT (this indicates that no one is watching television). Turn power ON. By moving slide switches 1 through 5, try various solutions to the problem. Light 4 will come ON only when the correct solution has been indicated.

BOOK IV

How Computers Do Arithmetic

1. THE BINARY NUMBER SYSTEM

In the previous books, we have frequently made use of the two-valued binary code. We have seen that this is a convenient representation for the on-off nature of the computer's components. Let us now consider the ***binary number system*** which is basic to the functioning of a computer system.

How Numbers Are Represented in the Binary System

All of us have been taught to think in the ***decimal number system***. The decimal system has ten different symbols—zero through nine—which can be used alone and in combination to represent numerical concepts. The binary system, however, uses only two different symbols—zero and one —to represent the same numerical concepts.

Let us consider a three-digit decimal number: for example, 547. We know that this actually means 500 plus 40 plus 7—or 5 times 100 plus 4 times 10 plus 7 times 1. In other words, we know that the right-most digit represents the number of 1's (units) in a number, the next digit to the left represents the number of 10's in the number, the next digit to the left represents the number of 100's in the number, and so on.

Now let us see why this is so. Suppose we start counting from 0. We go 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Now we have used up ***all*** of the symbols available in the decimal system. To continue, we must invent a method for handling numbers ***greater than 9***. We do this by placing a 1 to the left and a 0 in the far right position (called the "low order position") so that we now have "10" which represents one ten and no ones. Similarly, if we reach 99—which represents 9 tens and 9 ones—and then add one more we must invent a method for handling numbers larger than 99. We do this by placing a 1 to the left and 0's in the two low order positions, giving 100—1 times 100 plus 0 times 10 plus 0 times 1.

So we have a simple rule for forming any number system: when we run out of symbols in the low order position, we place a 1 in the next higher position and return the low order position to 0. Notice what this means: if we have ***10*** symbols, the low order position represents 1's; the next high order position represents 10's; the next high order position represents 100's; and so on. Each time we move one position to the left the "weighting" of that position is ten times as great as the "weighting" of the position to the right.

Now notice that if we had only ***5*** symbols—0, 1, 2, 3, 4,—to write the next higher number (5) we would have to place a 1 in the left position and a 0 in the right to represent one 5 and no 1's.

The binary system is actually simpler than the decimal system. We count 0, 1 and then we have run out of symbols. So the next symbol (to represent 2) must be 10 representing one 2 and no 1's. Three is written 11, representing one times 2 plus one times 1. Once again, we have run out of symbols. To write 4 we must write 100, representing 1 times 4 plus 0 times 2 plus 0 times 1.

Thus the rule for forming numbers in the binary system is: the low-order digit represents 1's, the next left digit represents 2's, the next left digit represents 4's, and so forth. Let us now count from 0 to 10 in binary numbers:

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

THE FIRST TEN BINARY NUMBERS AND THEIR DECIMAL EQUIVALENTS

Notice that each time we add 1 to a binary number—every time we count from one number to the next—the far right position ***changes***, either from a 0 to a 1, or from a 1 to a 0. And, if the right-most position is a 1 and 1 is added, the right-most position changes from a 1 to a 0 and the next left position ***also*** changes—either from a 1 to a 0 or from a 0 to a 1.

Since a computer works in the binary system, it must then be capable of ***changing***—either from a 1 to a 0, or from a 0 to a 1, if it is to perform arithmetic. We have already become acquainted with a program which permits the computer to do just this—the single-input flip-flop described in Book II.

Building a Single-Input Flip-Flop With Carry

However, to perform arithmetic the computer must also be able to "carry". That is, if the computer is to add 1 and 1 in binary, it must be able to indicate that the solution is: "1 and 1 are 0 and ***carry 1***". As a basic circuit to perform arithmetic then, we will need a "single-input flip-flop with carry".

In developing the single-input flip-flop in Book II, we first built a "two-input flip-flop" (see "Two-Input Memory Relay Circuit"—Book II.) This two-input flip-flop used only one relay which remembered either a zero or a one, depending upon which it had last been instructed to remember. To make this two-input flip-flop "change its mind," we had only to instruct it to remember the other number by pushing the appropriate pushbutton. However, to make the two-input flip-flop change, we have to know at what state it is in at this moment in order to make it change.

The single-input flip-flop, however, does the changing automatically by using a second relay to remember the state the flip-flop is currently in just long enough to allow the first relay (which is ***actually*** remembering the 1 or the 0) to change. By using a second relay, then, we no longer need to know what state the flip-flop is in at this moment in order to make it change. We merely instruct the computer to change, and it does so automatically because of the single-input flip-flop which is programmed onto it.

Let us go over again the functions of the two relays of the single-input flip-flop:

When the pushbutton is pushed, relay A instructs relay B to go to the same state relay A is in at this moment. When the pushbutton is released, relay B instructs relay A to move to the ***opposite*** state. Relay A does the actual remembering; relay B assists relay A when it is necessary for relay A to change.

To build a single-input flip-flop on the MINIVAC, we will begin with two relays wired to be self holding. (That is, each relay will remain in position once it is set.) The program for this is:

5+/5H	6G/6F
5H/6H	5C/6C
5G/5F	6C/6–

We will now ***add*** to the program so that when pushbutton 6 is pushed, relay 6 will go to the same state that relay 5 is in. This will require the following additional connections:

6H/6Y 5K/6F
6X/5L 5N/6E

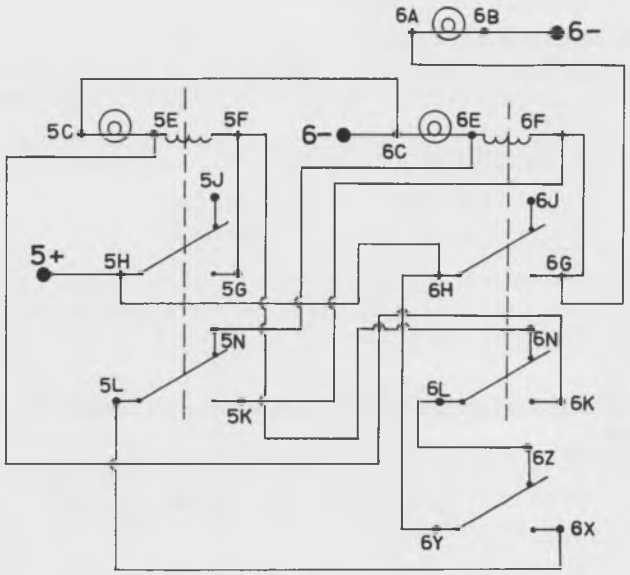
Again we will add to the program so that when pushbutton 6 is released, relay 5 will go to the ***opposite*** state from relay 6:

6Z/6L
6K/5E
6N/5F

And finally, we will add an indicator light to show what state the flip-flop is in:

6G/6A 6B/6—

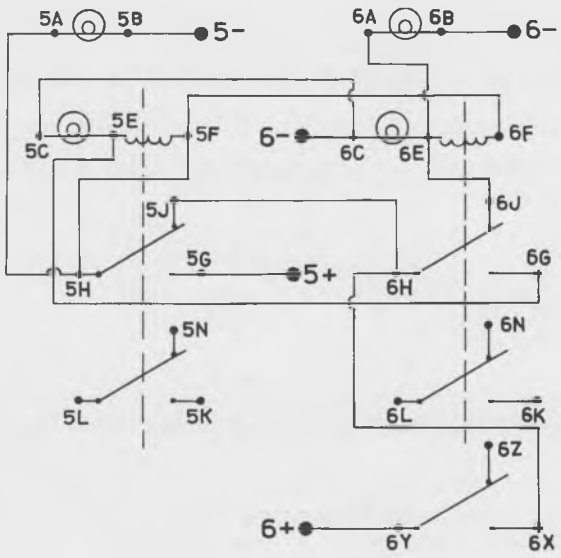
We now have a single-input-flip-flop programmed on MINIVAC. The circuit diagram for this is:



SINGLE-INPUT FLIP-FLOP CIRCUIT

However, our single-input flip-flop still cannot be used for arithmetic: it does not provide a "carry signal". That is, it does not notify us when it has changed from 1 to zero. And, since all contacts on both relays are used, we cannot arrange for a carry signal by merely adding connections.

The solution to this difficulty is to build a different version of the single-input flip-flop—one which leaves contacts free for the programming of a carry signal. The version of the flip-flop which we will use differs from the previous one in another respect as well. The stable states of the flip-flop when the pushbutton is up are either ***both*** relays ON or ***both*** relays OFF. For this program, light 5 is used to indicate the state of the flip-flop. Light 6 is used only to provide a path for current when relay 6 is "shorted-out" (OFF); light 6 is not used as an indicator in this program. The program and circuit diagram for this flip-flop are:



Program:

5+/5G	6J/6E
5H/5F	6E/6A
5F/6F	6B/6—
5C/6C	5H/5A
6C/6—	5B/5—
5J/6H	6+/6Y
6G/5E	6X/6H

SINGLE-INPUT FLIP-FLOP—SECOND VERSION

We now have a single-input flip-flop with contacts free for a carry signal. With the power OFF, program this flip-flop on your MINIVAC 601. Then:

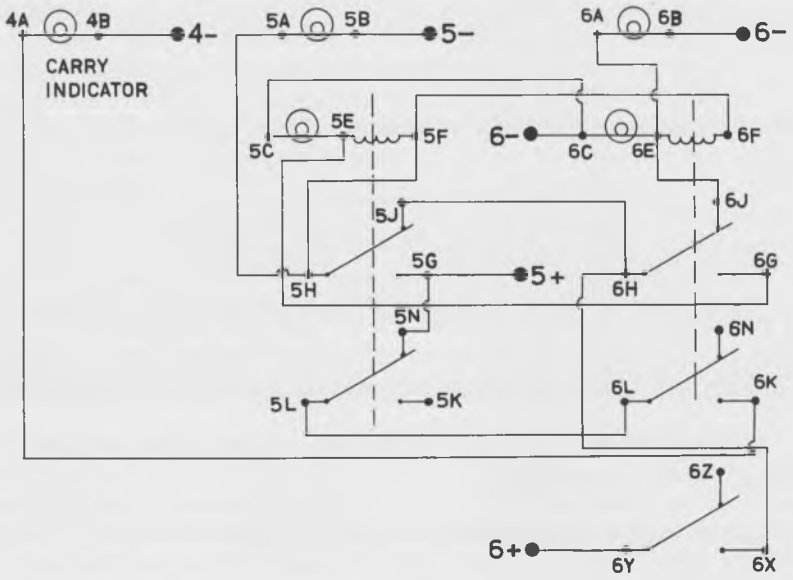
1. Turn power ON. Notice that ***both*** relays 5 and 6 are OFF.
2. Push and release pushbutton 6. Notice that ***both*** relays 5 and 6 are now ON.
3. Push pushbutton 6, but do not release. Notice that relay 5 is OFF; relay 6 is ON. We want a carry signal to occur NOW.

To add the carry signal, we will add the connections:

5G/5N	6K/4A
5L/6L	4B/4—

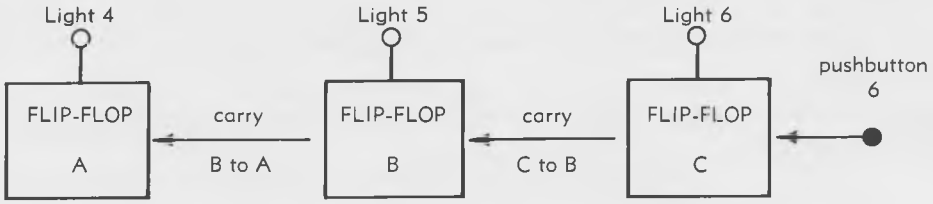
Light 4 will now indicate that a carry is taking place.

We now have a basic computer arithmetic device—the single-input flip-flop with carry. The circuit diagram looks like this:



CIRCUIT FOR SINGLE-INPUT FLIP-FLOP WITH CARRY

We can use single-input flip-flops as "building blocks" for more complex arithmetic devices. For example, three flip-flops can be wired together on your MINIVAC to produce a ***counter*** which operates as follows:



As pushbutton 6 is pushed and released again and again, this device will store in binary code the number of times the pushbutton has been pushed. The following chart shows what happens to each stage of the counter as the pushbutton is pushed and released, pushed and released: (the chart is read from right to left).

Flip-Flop A	Carry B to A	Flip-Flop B	Carry C to B	Flip-Flop C	Pushbutton carry	Times Pushed
0 0	NO NO	0 0	NO NO	0 1	UP DOWN	0
0 0	NO NO	0 1	NO YES	1 1	UP DOWN	1
0 0	NO NO	1 1	NO NO	0 1	UP DOWN	2
0 1	NO YES	1 1	NO YES	1 1	UP DOWN	3
1 1	NO NO	0 0	NO NO	0 1	UP DOWN	4
1 1	NO NO	1 1	NO YES	1 1	UP DOWN	5
1 1	NO NO	1 1	NO NO	0 1	UP DOWN	6
1 1	NO YES	1 1	NO YES	1 1	UP DOWN	7
0	NO	0	NO	0	UP	0 or 8

BINARY COUNTER—SEQUENCE OF OPERATIONS

This counter, shown in the preceding block diagram, can be built on the MINIVAC as the following experiment demonstrates:

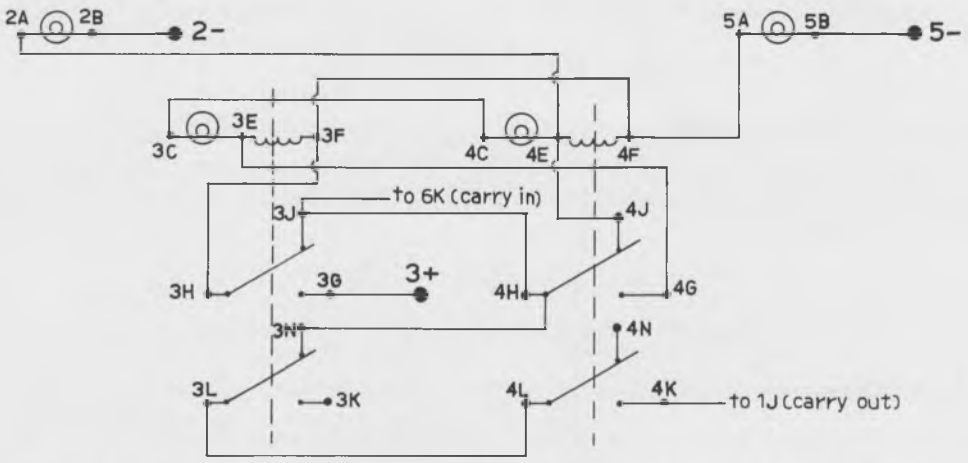
Experiment 1: A Three-Bit Binary Counter

In this Experiment we will build a ***counter*** and use it to see how the binary number system works.

A counter can be made up of any number of single-input flip-flops with carry. One flip-flop is required for each bit to be handled by the counter. Because the MINIVAC 601 has six relays, we

will be able to build a three-bit counter (using two relays for each flip-flop). This counter will allow us to count from 0 to 8.

The circuit diagram which follows is for the middle bit (or "2" bit) of the counter. The "high order" and "low order" bits (that is, the "4" bit and the "1" bit) are identical except that no "carry out" is required for the high order bit and the "carry in" to the low order bit is supplied through pushbutton 6. The program for the entire counter is given below with the circuit diagram for the middle stage:



CIRCUIT DIAGRAM FOR MIDDLE STAGE OF BINARY COUNTER

Full program for Binary Counter:

1A/2E	2C/3C	3J/6K	5F/5H
1B/1—	2E/2J	3L/4L	5G/5+
1C/2C	2F/4A	3N/4H	5J/6H
1E/2G	3A/6E	4B/4—	5J/6X
1F/2F	3B/3—	4C/5C	5L/6L
1F/1H	3C/4C	4E/4J	5N/6H
1G/1+	3E/4G	4F/5A	6A/6F
1J/2H	3F/3H	5B/5—	6B/6—
1J/4K	3F/4F	5C/6C	6C/6—
2A/4E	3G/3+	5E/6G	6E/6J
2B/2—	3J/4H	5F/6F	6Y/6+

Procedure:

1. Turn power ON. Push pushbutton 6 and release. Binary output lights 4, 5 and 6 will read OFF, OFF, ON or 001. The Binary Counter is displaying a 1.
2. Push pushbutton 6 again and release. Binary output lights 4, 5 and 6 will now read OFF, ON, OFF or 010. The Binary Counter is now displaying a 2.
3. Push and release pushbutton 6 several times. Each time after releasing the pushbutton, note the binary number displayed in lights 4, 5 and 6. Compare your numbers with the table of binary numbers on page 66.

You will notice that the counter repeats after it has reached 7. This is a result of the fact that the counter is made up of 3 flip-flops. Each flip-flop has 2 stable states (0 or 1), so the combination of 3 flip-flops has 2^3 or 8 stable states. These 8 stable states represent the binary numbers 0 through 7.

As you push pushbutton 6 several times, notice that the low order flip-flop (represented by light 6) changes each time the pushbutton is pushed and released. The low order flip-flop represents 1's (units). The next flip-flop to the left (represented by light 5) changes every other time the pushbutton is pushed and released. This flip-flop represents 2's. And of course, the high order

flip-flop (represented by light 4) changes every fourth time the pushbutton is pushed and released; the high order flip-flop thus represents 4's.

You will notice that the middle flip-flop changes on a regular basis; it changes every time the low order flip-flop changes from 1 to 0. The high order flip-flop behaves similarly—it changes every time the middle flip-flop changes from 1 to 0.

Referring again to the table of binary numbers, we can see that this is a characteristic of binary numbers: adding 1 to a binary number changes the low order bit—either from 0 to 1 or from 1 to 0. All other, higher order, bits change if and only if the bit to the right is changing from 1 to 0.

Leave the Binary Counter on the MINIVAC for Experiment 2.

Experiment 2: Counter Arithmetic

It is possible to perform additions very simply on a Binary Counter. We have already seen that it is possible to add 1 to the number in the counter merely by pushing and releasing pushbutton 6. This technique can easily be extended to any number which can be represented on the counter.

For example, to add 2 and 3: push and release pushbutton 6 twice; the counter will display the binary number 2. Now push the pushbutton 3 more times and the counter will display the answer—the binary number 5. This is exactly what would have happened had you just pushed the pushbutton 5 times.

This is one way in which an addition can be performed in a computer and is, in fact, a common method for handling certain kinds of addition problems.

At this point your counter should be displaying binary 5. Now push button 6 five more times. The pushbutton has now been pushed ten times, and the counter ***should*** display binary equivalent of decimal 10. However, binary 10 is a four-bit number (1010). Since we have only three bits in our counter, we would expect to see only the low order three bits of the number displayed. The bit pattern for the binary equivalent of decimal 10 is 1010. The counter will display 010.

Experiment 3: Universal Counter Arithmetic

It is possible to wire the Binary Counter so that it will count ***down*** as well as ***up***. In this way, both additions and subtractions can be performed. The program below gives the necessary modifications to the Binary Counter from experiments 1 and 2:

Modify the program from Experiment 1 as follows:

Remove the connections:

6K/3J
4K/1J

Add the connections:

3K/6W	5N/6N
3N/4N	5K/6T
6R/6K	6S/3J
4K/6U	6V/1J

Procedure:

1. Build the counter or, if you have the counter from experiments 1 and 2 programmed, make the necessary modifications. If slide switch 6 is LEFT, the counter will count UP; if slide switch 6 is RIGHT, the counter will count DOWN.
2. Subtract 4 from 7 as follows:
   a. Move slide switch 6 to the LEFT.
   b. Enter the larger number (7) by pushing pushbutton 6 the correct number of times. The counter is now displaying the larger number.

c. Move slide switch 6 to the RIGHT.
d. Enter the smaller number (4) by pushing pushbutton 6. The counter is now displaying the answer: $7 - 4 = 3$.

Note: addition is carried out exactly as in experiment 2 when slide switch 6 is LEFT.

2. BINARY ADDITION

Rules for Binary Addition

Let us add together two binary numbers: 11010 (26) and 11100 (28):

```
 11010
 11100
-------
```

starting from the low order (right-most) bit:

0 plus 0 equals 0.
1 plus 0 equals 1.
0 plus 1 equals 1.
1 plus 1 equals 0 and carry 1.
1 plus 1 plus carry 1 equals 1 and carry 1.
The answer is:

```
 11010
 11100
-------
110110
```

As you can see, the addition tables for binary arithmetic are very simple. They are:

```
 0    0    1    1
+0   +1   +0   +1
--   --   --   --
 0    1    1   10
```

The problem above illustrates how a binary adder for a computer must work. Each bit of the adder must be able to accept three inputs:

a. the first number to be added
b. the second number to be added.
c. a carry from the bit to the right.

Each bit of the adder must also be able to yield two outputs:

a. the sum
b. the carry.

Let us call the three inputs "A", "B", and "carry in"; and let us call the two outputs "sum" and "carry out". Notice that the ***sum*** will equal 1 if and only if either one or three of the inputs is 1. ***Carry out*** will equal 1 if and only if either two or three of the inputs are 1.

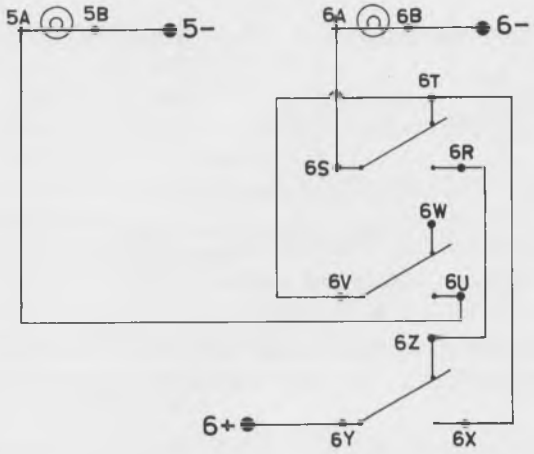
If we restrict the input to only A and B, we can build the following table showing inputs and outputs of an adder:

A	B	Carry Out	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The circuit which actually performs this operation is called a "half-adder with carry". A half-adder with carry is a device with two inputs and two outputs such that the sum will equal 1 if and only if one and only one of the inputs is 1. In other words, the sum circuit is an EITHER BUT NOT BOTH circuit. The following experiment illustrates the operation of a half-adder with carry.

Experiment 4: A Half-Adder with Carry

In this experiment we will build and test a simple half-adder with carry. The circuit and program are:



Program:

5A/6U	6R/6Z
5B/5—	6T/6X
6A/6S	6T/6V
6B/6—	6Y/6+

The entire program requires one pushbutton, one slide switch and two lights. The slide switch represents input A; the pushbutton represents input B. The lights represent the two outputs: light 6 represents the ***sum;*** light 5 represents ***carry out***.

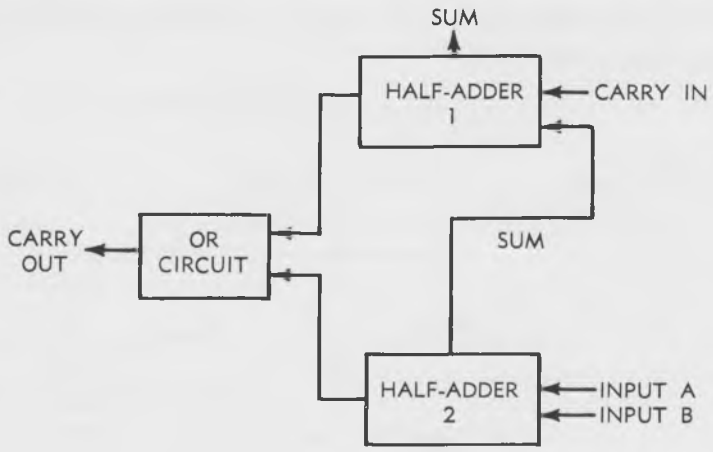
Procedure:

1. Turn power ON. Move slide switch 6 to the RIGHT: input A is now 0. Do ***not*** push pushbutton 6: input B is 0. Light 5 and 6 are OFF: sum is 0 and carry out is 0.
2. Push pushbutton 6 and hold it down. Input B is now 1. Light 6 comes ON, indicating that the sum is 1. This is equivalent to the mathematical statement: $0 + 1 = 1$.
3. Release pushbutton 6 and move slide switch 6 to the LEFT. Input A is now 1; input B is 0. Light 6 comes ON, indicating that the sum is 1: $1 + 0 = 1$.
4. With the slide switch LEFT, push pushbutton 6. Light 6 goes OFF and light 5 comes ON. This represents the mathematical statement: $1 + 1 = 0$ and carry 1.

If we now have three inputs (A, B, and carry in) and two outputs, the table of values looks like this:

A	B	Carry In	Carry Out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

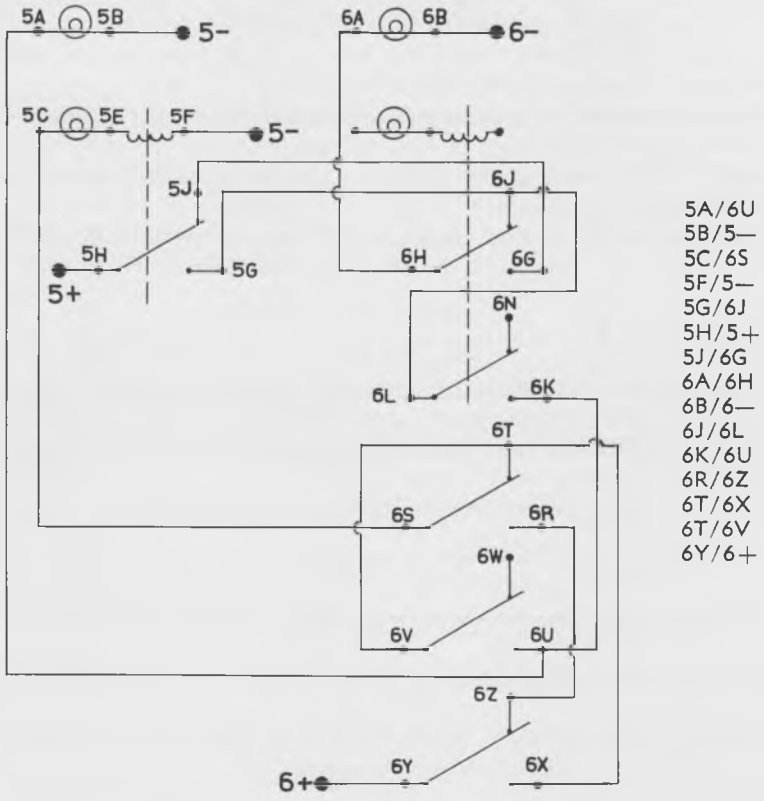
The circuit which actually performs this operation is called a "full adder with carry", and is made up of two half-adders as follows:



Like the single-input flip-flop with carry, the half-adder with carry is a basic computer circuit. Half-adders can be used as "building blocks" for more complex arithmetic devices—an example of which is the full adder.

Experiment 5: A Full Adder

In this experiment we will modify the circuit of experiment 4 to build a full adder. A full adder is made up of two half-adders and a simple OR circuit. The circuit and program for a full adder are:



Relay 6 represents a carry in from a previous stage. Relay 5 represents the sum of inputs A and B; this sum will be added to the carry in to give the sum for the full adder. Notice that a carry signal may come from ***either*** of two places: either from the half-adder representing the slide switch and pushbutton (input A and input B), or from the half-adder representing the carry in (relay 6) and the sum of the input (relay 5). Either half-adder can produce a carry signal.

Procedure:

1. Turn power ON. Move slide switch 6 to the RIGHT; Input A is 0. Do ***not*** push pushbutton 6; input B is 0. Relay 6 is OFF; carry in is 0. Lights 5 and 6 are OFF. This is equivalent to the mathematical expression: $0 + 0 + 0 = 0$.
2. Move slide switch 6 to the LEFT; input A is now 1. Light 6 comes ON: $1 + 0 + 0 = 1$.
3. Manually move relay 6 to the left to indicate that carry in is 1. Light 6 goes OFF and light 5 comes ON: $1 + 0 + 1 = 0$ and carry 1.
4. Try other combinations of inputs by moving slide switch 6, pushing pushbutton 6, and/or manually moving relay 6.

We have now seen how two half-adders can be combined to produce a full adder. Full adders can in turn be used as building blocks for adders capable of accepting greater inputs and yielding greater outputs. The following experiment illustrates the combination of three full adders to produce a "three-bit adder".

Experiment 6: A Three-Bit Adder

In this experiment we will build a device which is capable of adding together two 3-bit binary numbers and displaying a result of not more than 4 binary bits. The program for the 3-bit adder is:

2C/3G	3L/4X	5K/6W
2F/3F	4A/4V	5L/6A
2G/4S	4B/4—	5N/6U
2H/4Y	4E/5F	5R/6T
2K/4W	4F/5S	5T/6R
2L/2+	4G/5U	5T/5V
2N/4U	4H/4N	5Y/6V
3A/M10	4K/5Z	6B/6—
3B/3—	4L/5A	6E/6—
3C/4G	4N/5X	6G/6R
3F/4E	4Y/5Y	6H/6X
3G/3N	5B/5—	6J/6K
3H/4Z	5C/6K	6L/6Z
3J/4T	5F/6E	6N/6T
3K/4R	5G/6S	6V/6Y
3K/M10	5H/5+	6Y/6+

The three full adders which make up this circuit are very much like the single adder of experiment 5. However, the circuit has been modified slightly so that three adders use only five relays. The high order stage of the adder is wired so that the carry from it is displayed by light 3. This allows us to display a 4-bit result with a 3-bit adder.

Procedure:

1. Turn power ON. Enter a binary number in the adder using slide switches 4, 5 and 6. For example, to enter binary 4 (100) set slide switch 4 LEFT, slide switch 5 RIGHT and slide switch 6 RIGHT.
2. Enter another binary number using pushbuttons 4, 5 and 6. For example, to enter binary 5 (101) push button 4 and 6; do not push pushbutton 5.
3. The answer will be displayed in binary output lights 3, 4, 5 and 6. For example, if lights 3 and 6 are ON and lights 4 and 5 are OFF, the answer is 1001 (9).

3. HOW COMPUTERS SUBTRACT

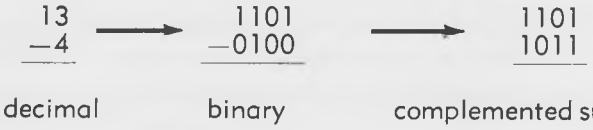
We have now discussed two methods of computer addition. The first method was a counter which was dependent upon the number of times an input button had been pushed. The second method used three full adders with carry which allowed us to enter two 3-bit numbers simultaneously.

Two's Complement Arithmetic

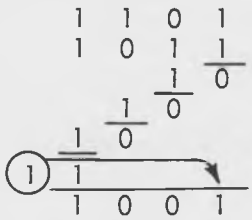
Similarly, there are two methods of computer subtraction. The simpler method is the "down counter" demonstrated in experiment 3. However, in spite of its simplicity, the down counter is seldom used in a computer. The more common method of subtraction used in computers is called "two's complement arithmetic". Two's complement arithmetic in the binary system is the same as ten's complement arithmetic in the decimal system.

Two's complement arithmetic in the binary system works as follows: the subtractor is "complemented". This means that every 1 is replaced by a 0, and every 0 is replaced by a 1. The complemented subtractor and the subtrahend are then ***added***. If a carry occurs out of the high order position, it is added to the low order position. The resulting sum is the proper difference between the two numbers.

As an example, let us subtract binary 4 from binary 13. We first write down 1101 (13) and below it we write 1011—which is the ***complement*** of 0100 (4):



We now add in the usual binary manner except that if a carry occurs out of the high order bit, we will add it to the low order bit:



A great advantage of two's complement arithemetic is the fact that there is an automatic bookkeeping method for indicating the sign of the result. If a carry occurs out of the high order position, the subtraction has resulted in a positive number. On the other hand, if the carry does not occur, the result is a negative number.

The experiment which follows illustrates the operation of two's complement subtraction on the MINIVAC.

Experiment 7. A Three-Bit Subtractor

In this experiment we will program the MINIVAC to perform subtraction by the two's complement method. This will be done by modifying the three-bit adder of experiment 6 as follows:

The ***carry out*** from the high order stage (represented by the wire connected to light 3) will be connected to the ***carry in*** of the low order stage. In this way, if a high order carry occurs it will be read into the low order stage, just as was illustrated above.

The adder will also be modified so that the subtractor is automatically complemented. To build the three-bit subtractor, make the following changes to the program from experiment 6:

REMOVE these connections:

3A/M10	4K/5Z
3L/4X	6H/6X
3H/4Z	6L/6Z
4N/5X	

Now ADD these connections:

6F/M10	4K/5X
3L/4Z	6C/6F
3H/4X	6H/6Z
4N/5Z	6L/6X

To use the program:
The subtractor is used in the same manner as was the adder.
A three-bit binary number (the subtrahend) is entered on slide switches 4, 5, and 6. The subtractor is then entered on pushbutton 4, 5 and 6. The answer will be displayed in the binary output lights.

If we attempt to subtract a larger number from a smaller number, we will have an incorrect answer. This condition can be detected by observing the carry in relay of the low order stage. If this relay fails to pull in when the subtraction takes place, we have attempted an incorrect subtraction.
An example:
To subtract binary 2 (010) from binary 5(101):
1. Enter the subtrahend (101) by moving slide switches 4 and 6 to the LEFT, leaving slide switch 5 RIGHT.
2. Enter the subtractor (010) by pushing pushbutton 5 DOWN and leaving pushbuttons 4 and 6 UP. We need not enter the complemented subtractor; the program automatically complements the subtractor.
3. The answer appears in the binary output lights: light 4 is OFF, and lights 5 and 6 are ON. This is read as 011, which is binary 3 : $5 - 2 = 3$.
Had we attempted to enter binary 7 (111) in the pushbuttons, relay 6—the carry in relay for the low order stage— would not have pulled in and the answer displayed in the output lights would be incorrect.

4. COMPUTER MULTIPLICATION

Binary Multiplication

In the decimal number system we can multiply by ten merely by placing a zero to the right of the number. Similarly, in the binary system we can multiply by ***two*** by placing a zero to the right of the number.

To multiply in the binary system, we need only remember that:

$0 \times 0 = 0$	$0 \times 1 = 0$
$1 \times 0 = 0$	$1 \times 1 = 1$

and:

10 (2) X 1 = 10 (2)
100 (4) X 1 = 100 (4)
1000 (8) X 1 = 1000 (8), etc.

In order to multiply in the binary system on a computer we will need a device known as a ***shift register.*** A shift register ***shifts*** each bit of a number one space to the left which, of course, rep-

resents multiplication by two. Multiplication by 4 is accomplished by shifting left twice; multiplication by 8 is accomplished by shifting left three times, and so on.

The following experiment demonstrates how a computer can multiply using a manual ***shift register.***

Experiment 8: The Shifting Operation

In this experiment we will multiply and divide numbers by powers of two by manually shifting the numbers right and left. The device which we will use for this operation is called an ***accumulator.*** In this experiment, we will be concerned only with its shifting abilities.

The program for the accumulator is:

1A/1S	2H/4Y	3T/4R	5L/6A
1B/1—	2K/4W	4A/4V	5N/6U
1C/3U	2L/2+	4B/4—	5R/6T
1F/2F	2L/2S	4E/5F	5T/6R
1G/2W	2N/4U	4F/5S	5T/5V
1H/2A	2S/2V	4G/5U	5Y/6V
1J/2U	3A/3S	4H/4N	6B/6—
1K/1T	3B/3—	4K/5Z	6C/M10
1L/2R	3C/4G	4L/5A	6E/6—
1N/1R	3F/4E	4N/5X	6G/6R
1N/2T	3G/3N	4Y/5Y	6H/6X
1U/M10	3H/4Z	5B/5—	6J/6K
1V/1T	3J/3R	5C/6K	6L/6Z
2B/2—	3J/4T	5F/6E	6N/6T
2C/3G	3K/4R	5G/6S	6V/6Y
2F/3F	3L/4X	5H/5+	6Y/6+
2G/4S	3V/3T	5K/6W	

To multiply by two, we will perform a ***left shift operation*** as follows:

1. Enter a binary number by setting the slide switches appropriately. (A slide switch RIGHT represents 0; a slide switch LEFT represents 1.) The number is now displayed in the binary output lights.
2. Beginning at the ***left*** end of the console, set slide switch 1 to whatever position slide switch 2 is in; set slide switch 2 to whatever position slide switch 3 is in; set slide switch 3 to whatever position slide switch 4 is in. Continue until slide switches 1 through 5 are in the positions that slide switches 2 through 6 were previously in. Set slide switch 6 to zero. The result will be displayed in the binary output lights.

To multiply by four, enter a binary number on the slide switches and perform ***two consecutive*** left shift operations. To multiply by eight, enter a three-bit binary number in slide switches 4, 5 and 6 and perform ***three consecutive*** left shift operations.

To divide by two, we will perform a ***right shift operation*** as follows:

1. Enter a binary number by setting the slide switches appropriately. (Note: it is usually more convenient to move the ***binary point***—which corresponds to the ***decimal point***—to the left for division. For instance, we might place the binary point ***between*** slide switches 3 and 4, in which case the dividend would be entered on slide switches, 1, 2 and 3.)
2. Beginning at the ***right*** end of the console, set slide switch 6 to whatever position slide switch 5 is in; set slide switch 5 to whatever position slide switch 4 is in. Continue until slide switches 2 through 6 are in the positions slide switches 1 through 5 were previously in. Set slide switch 1 to zero. The result will be displayed in the binary output lights. Anything to the ***right*** of the binary point is ***remainder.***

To divide by four, enter a binary number and perform ***two consecutive*** right shift operations.

To divide by eight, enter a binary number and perform ***three consecutive*** right shift operations.

This manual shift register demonstrates multiplication and division by powers of two on an accumulator. Large digital computers perform shifts exactly like these, but the shifting is done automatically. (See appendix to Book IV: Automatic Shift Register).

Leave this program wired on your MINIVAC for experiment 9.

Multiplication by Numbers other than Powers of Two

Multiplication in the binary system can be done exactly as it is in the decimal system by multiplying by each bit of the multiplier and adding. For example:

$$\begin{array}{r} 1011 \\ \times\ \ 11 \\ \hline 1011 \\ 1011\ \ \\ \hline 100001 \end{array}$$

This same operation can also be done in a slightly different manner. For instance, let us multiply 111 (7) X 110 (6)

Let:

111 = Multiplicand
110 = Multiplier
Then: 111 X 110 = ?

First, we multiply the multiplicand by the ***left-most*** bit of the multiplier:

$$111 \times 1 = 111$$

Now, we shift left one by adding a 0 to the right:

$$1110$$

To this, we add the product of the multiplicand and the second left-most bit of the multiplier:

$$\begin{array}{r} 1110 \\ 111 \times 1 = 111 \\ \hline 10101 \end{array}$$

Once again, we shift left one by adding a 0 to the right:
101010
Again, we add the product of the multiplicand and the next left-most bit of the multiplier:

$$\begin{array}{r} 101010 \\ 111 \times 0 = \ \ 000 \\ \hline 101010 \end{array}$$

which is the final result:

$$111 \times 110 = 101010 \text{ or } 7 \times 6 = 42$$

Multiplication on a computer is carried out in just this fashion, using an ***accumulator.*** The accumulator acts just like a piece of paper on which we record the steps which were illustrated above. The following experiment demonstrates multiplication on a computer using an accumulator.

Experiment 9: The Accumulator

Without changing the program from experiment 8 we will now perform multiplication by numbers other than powers of two. Multiplications will be performed by successive additions and shifts.

The ***accumulator,*** which is wired on the console from experiment 8, is a device which is capable of performing ***both*** additions and shifts.

As an example, le us multiply:

6 X 10 or in binary: 110 X 1010.
Let: 110 = Multiplicand
1010 = Multiplier

The left-most bit of the multiplier is 1, so we will enter into the accumulator:

110 X 1 or 110.

We will do this by pushing the appropriate pushbuttons (in this case, pushbuttons 4 and 5.) The slide switches should all be RIGHT.
As we push the pushbuttons, the output lights display the number—000110. Record this number on a piece of paper and release the pushbuttons.
Enter the number which you recorded in the slide switches and perform a left shift one bit as described in experiment 8. The output lights will now read 001100.
The next left-most bit of the multiplier is 0. We will add nothing to the accumulator, but will shift left one bit. The output lights now show 011000.
The next left-most bit of the multiplier is 1, so we will add—through the pushbuttons:

110 X 1 or 110

The output lights now show 011110. Record this number on a piece of paper and release the pushbuttons. Enter the number which you just recorded in the slide switches and shift left one bit. The output lights now show 11100.
The low order bit of the multiplier is 0, so it is not necessary to add anything more into the accumulator. The binary output lights now show the final result: 11100 or 60.

110 X 1010 = 111100 or 6 X 10 = 60

Select other numbers and multiply them on the accumulator. The limit of this accumulator is 111111 (63).

5. DIVISION ON A COMPUTER

Binary Division

Division in the binary system is performed exactly as in the decimal system. For example, to divide 11011 (27) by 101 (5):

```
        101
    ------
101 | 11011
      101
      ---
       111
       101
       ---
        10
```

11011 ÷ 101 = 101 plus a remainder of 10
or
27 ÷ 5 = 5 plus a remainder of 2.

Performing this sequence of operations on a computer can, however, be difficult. In fact, many computers have no provision for automatic division. For such a computer, a ***subroutine*** is written so that the computer can carry out the individual steps of a division. That is, the computer is given a program of instructions which, when followed, permit the computer to divide.

The following experiment demonstrates the steps involved in certain types of computer division. The actual program required is a combination shift register and subtractor.

Experiment 10: Division

This experiment demonstrates division on the MINIVAC with your help. The program and circuit diagram for a combination shift register and subtractor which will allow us to divide are:

1A/1V	2F/3F	3G/3N	4H/4N
1B/1—	2L/2+	3K/M10	4S/4+
1R/3K	2R/3T	3R/6G	5A/5R
1S/2G	2S/4F	3S/5G	5B/5—
1T/3J	2T/3R	3T/6N	5C/6K
1U/2N	2T/2V	3U/5N	5F/6E
1W/2K	2U/4G	3V/3Y	5H/5+
1X/3H	2X/4K	3W/5K	5S/5+
1Y/2H	2Y/6S	3X/6L	6A/6R
1Y/2Y	2Z/4N	3Y/3+	6B/6—
1Z/3L	3A/5L	3Z/6H	6E/6—
2A/4L	3B/3—	4A/4R	6F/M10
2B/2—	3C/4G	4B/4—	6J/6K
2C/3G	3F/4E	4E/5F	6S/6+

We will now divide 011011 (27) by 101 (5) as follows:
Write the dividend on a piece of paper, with a vertical line between the third and fourth digits. The vertical line is always placed so that the divisor is greater than whatever is to the left of the line. Now write the divisor under the left end of the dividend:

```
011 | 011
----+----
101 |
```

Place the dividend on the slide switches. The dividend is now displayed in the output lights as 011011.
The divisor is greater than the left-most three bits of the dividend. Therefore, shift left one bit.
We now have:

```
110 | 110
----+----
101 |
```

Subtract the divisor from the dividend by entering the divisor on pushbuttons 1, 2 and 3. The result will appear in the output lights:

```
110 | 110
----+----
101 |
====+====
001 | 110
```

Place this result in the slide switches. Since the subtraction proceeded properly, we will add 1 to the low order bit and place this new result in the slide switches:

001	111

To go on to the next step, shift left one bit and again write the divisor below the left-most three bits:

011	110
101	

The divisor is greater than the left-most three bits. Therefore, we will shift left one bit:

111	100
101	

We can now subtract by entering the divisor in pushbuttons 1, 2 and 3 and entering the result on the slide switches:

111	100
101	
010	100

Since the subtraction proceeded properly, we will add 1 to the low order bit to yield:

010	101

The division has now been completed: There have been as many shifts left as there were bits to the right of the vertical line. The answer is to the right of the vertical line; the remainder is to the left of the vertical line:

remainder	answer
010	101

11011 ÷ 101 = 101 plus a remainder of 10.

or

27 ÷ 5 = 5 plus a remainder of 2.

6. CONVERSIONS

Since computers work in the binary system and we generally work in the decimal system, an important function of a computer is conversion between the two systems. The following experiments give the programs for conversions between the binary and decimal systems.

Experiment 11. Decimal to Binary Converter

Conversion from decimal to binary is a particularly important function for input devices. There are two methods for conversion commonly in use in large digital computers. The first method is a ***counting*** scheme which is particularly suitable for conversion of numbers on a punched card. With this method the computer generates a binary number on the basis of the length of time which elapses before a decimal code is observed.

As an example of how the counting method works, consider the following:

A card is fed through a card reader at a constant rate of speed. When the edge of the card is detected in the reader, the counter in the computer comes on. When a punch in the card is detected, the counter goes off and a binary number corresponding to the decimal number punched on the card is displayed in the counter. This method depends on the time lapse between the card edge and the punch.

It is possible to build a converter of this type for the MINIVAC, using the rotary switch to actuate a counter. Since you are familiar with the rotary switch mechanism and a counter, you will be able to build such a device yourself.

The second method for conversion from decimal to binary is essentially a ***decoding*** scheme. In this method, the computer is programmed to recognize a decimal number by the location of the input pulse; a series of direct contacts then convert this pulse to the appropriate binary number.

The following program mechanizes such a method. Setting the rotary switch to any number from 0 through 10 will cause the equivalent binary number to appear in the output lights.

Program for Decimal to Binary Converter:

1E/D10	3B/3—	4H/4L	6B/6—
1F/2F	3E/D7	4H/5H	6E/D3
1G/3com	3F/4F	4K/4com	6F/6—
1H/1L	3G/6E	5A/5com	6G/6com
1K/6K	3H/3L	5B/5—	6H/6L
1L/2L	3K/4K	5E/D5	6H/6+
2E/D9	3L/4L	5F/6F	6K/5com
2F/3F	4A/4com	5G/6G	D1/6com
2G/5G	4B/4—	5H/5L	D2/5com
2H/2L	4E/D6	5K/4com	D4/4com
2H/3H	4F/5F	5L/6L	D8/3com
2K/3com	4G/5com	6A/6com	D16/M+
3A/3com			

To use the Decimal to Binary Converter:

Enter any decimal number from 0 through 10 by turning the rotary switch to the desired position. The binary equivalent of the decimal number will appear in the binary output lights. (A light ON represents 1; a light OFF represents 0.)

Experiment 12: Binary to Decimal Converter

Conversion from binary to decimal is generally handled by a "tree" circuit (For an example of a "tree" circuit, see the Mind Reading Program of Book III.)

With a "tree" circuit, each combination of switches energizes one and only one terminal of the rotary switch. The rotary switch is programmed so that when it turns to the energized terminal it will stop. Thus, setting a binary number in the slide switches will result in the same number being displayed on the rotary switch.

Program for Binary to Decimal Converter:

1C/4R	2K/3L	4C/5C	5K/D1
1F/2F	2L/4W	4F/5F	5N/D0
1G/6H	2N/4L	4G/D13	5S/6S
1H/5U	3C/6R	4J/D12	6F/6—
1J/5H	3C/4C	4K/D9	6G/D7
1K/6L	3F/4F	4N/D8	6J/D6
1L/5W	3G/D15	4S/4+	6K/D5
1N/5L	3J/D14	4S/5S	6N/D4
2C/5R	3K/D11	5C/6C	6X/D17
2F/3F	3N/D10	5F/6F	6Y/6+
2G/3H	3U/4V	5G/D3	D16/D19
2H/4U	3V/3—	5J/D2	D18/M—
2J/4H	3W/5V		

To use the Binary to Decimal Converter:

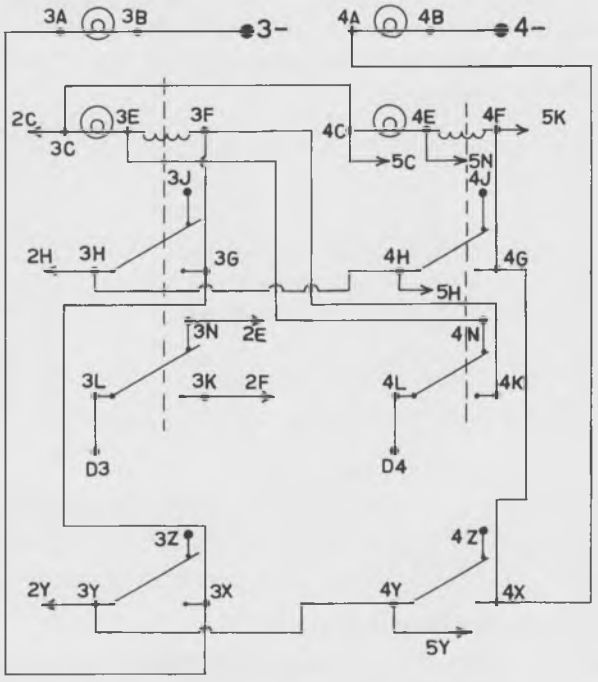
Enter any binary number from 0 through 15 on slide switches 3 through 6. Push pushbutton 6. The rotary switch will indicate the decimal equivalent of the binary number.

APPENDIX

Automatic Shift Register

The shift register which was previously discussed can be "automated" so that the computer will carry out the instruction: SHIFT LEFT ONE BIT whenever the rotary switch dial is turned. Although this is a more convenient method of shifting-as opposed to manually operating the slide switches—you can see that it uses many more contacts.

The program for the entire six-bit automatic shift register follows, with a circuit diagram for the middle two bits:



CIRCUIT DIAGRAM: MIDDLE TWO BITS OF AUTOMATIC SHIFT REGISTER

Program for full six-bit automatic shift register:

1A/1X	3A/3X	5A/5X
1B/1—	3B/3—	5B/5—
1C/2C	3C/4C	5C/6C
1E/2N	3E/4N	5E/6N
1F/1G	3F/3G	5F/5G
1F/2K	3F/4K	5F/6K
1G/1X	3G/3X	5G/5X
1H/2H	3H/4H	5H/6H
1L/D1	3L/D3	5L/D5
1Y/2Y	3Y/4Y	5Y/6Y
2A/2X	4A/4X	6A/6X
2B/2—	4B/4—	6B/6—
2C/3C	4C/5C	6C/6—
2E/3N	4E/5N	6E/D7
2F/2G	4F/4G	6F/6G
2F/3K	4F/5K	6G/6X
2G/2X	4G/4X	6H/6+
2H/3H	4H/5H	6L/D6
2L/D2	4L/D4	6Y/D16
2Y/3Y	4Y/5Y	D16/M+

To use the automatic shift register:

Enter a binary number in the shift register by pushing and releasing the appropriate pushbuttons. The number will appear in the output lights.

Shift LEFT one bit by slowly turning the rotary switch dial one complete revolution. The shifted binary number will now appear in the output lights.

To shift left more than one bit simply turn the rotary switch dial the desired number of revolutions.

Two-Bit Adder with Automatic Decimal Conversion

The following program is given as an example of the combination of basic computer arithmetic circuits. A two-bit adder is combined with binary-to-decimal converter so that when two 2-bit binary numbers are added together the sum is shown on the decimal output dial.
The program for the two-bit adder with automatic decimal conversion is:

1C/4H	3J/D3	5U/5Z
1F/2F	3K/D6	5W/5X
1G/3H	3N/D2	5Y/6Y
1H/6G	4A/5R	6A/6C
1J/2H	4A/4K	6B/6—
1K/3L	4B/4—	6C/6V
1L/6J	4C/5V	6F/6—
1N/2L	4F/5F	6H/5—
2C/4K	4G/5J	6S/6W
2C/3C	4H/5A	6U/6Z
2F/3F	4J/5G	6W/6X
2G/D5	4L/4J	6Y/6+
2J/D1	5B/5—	D16/D19
2K/D4	5C/6R	D17/M+
2N/DO	5F/6F	D18/M—
3F/4F	5H/5+	
3G/D7	5S/5W	

To use the program:

Enter a 2-bit binary number in slide switches 5 and 6.

Enter a second 2-bit binary number on pushbuttons 5 and 6.

The rotary switch dial will turn to the decimal sum of the numbers. (Binary output lights 4, 5 and 6 will display the sum in binary.)

MINIVAC 601 AND THE MINIVAC MANUAL

ARE PRODUCTS OF:

372 Main Street, Watertown, Massachusetts