Computer Systems Department

BUIC III COMPUTER PRINCIPLES

27 December 1967

Keesler Technical Training Center
Keesler Air Force Base, Mississippi

Computer Systems Department
Keesler AFB, Mississippi

STUDENT TEXT 2OSR0123-3 C 179-BUIC-ST
27 December 1967

BUIC III COMPUTER PRINCIPLES

This student text provides study and reference material in support of Block I, Course 2OSR0123-3.

CONTENTS

	Title	Page
INTRODUCTION:	COMPUTERS, THE HEART OF AUTOMATION	1
CHAPTER 1.	INTRODUCTION TO COMPUTERS	4
	History of Computers	4
	Basic Classes of Computing Machines	7
	Digital Computer Elements	8
CHAPTER 2.	COMPUTER MATHEMATICS	12
	Number Systems	12
	Numbering System Conversions	15
	Arithmetic Operations	17
CHAPTER 3.	DATA REPRESENTATION	25
	General	25
	Coded Method	25
	Hollerith Coding	26
	Numeric Representation	28
CHAPTER 4.	BASIC COMPUTER LOGIC	30
	Information Signals	30
	Switching Logic	32
	Computer Logic Circuits	35
	Small-Scale Storage Circuits	37
	Arithmetic and Control	39
CHAPTER 5.	COMPUTER SYSTEMS	44
	Large-Scale Storage and Memory	44
	Description of BUIC III Equipment	54
	Flow of Information Within the Computer	57
	BUIC III Equipment Configuration	58

INTRODUCTION

COMPUTERS, THE HEART OF AUTOMATION

by

Dr. John R. Pierce

Executive Director, Research-Communications Sciences Division

Bell Telephone Laboratories, Inc.

The general purpose digital computer or a specialized relative, is the heart, or rather the brain, of automation. Whether it is in a computing center or hitched to an airplane, a train, a telephone switching system, a machine or a factory; whether its output is numbers, or pictures, or words that may write orders, pay employees, or help the boss; or whether the output is physical action, the computer provides the "auto" in "automation."

When I started work in electronics in 1936, all we had to help us make calculations were slide rules and desk calculating machines. By punching the buttons of a calculating machine one could add, subtract, multiply and divide numbers. With a little ingenuity one could take square roots and do other mathematical operations. But one had to write down the outcome of each operation and do many operations in succession before a complicated computing job was finally completed. This was slow as well as boring, and one did not do much of it.

Today, electronics, communication, and computation are inextricably mingled; and they are mixed up with transportation and production as well. There are 22,500 general purpose electronic digital computers in schools, laboratories, offices, and factories.

Today computers calculate, keep books, keep track of inventories and order stock, handle airline and railway reservations, control railroad traffic, control machine tools and chemical processing plants, justify and hyphenate newspaper test, make indexes of of scientific papers and journals, guide rockets and control the launching of satellites, and do a host of other complicated but clearly defined jobs that are important to industry, science, and government. It is widespread industrial and government usage, rather than scientific calculation, that accounts for a five billion dollar production and sales of computers in 1964, and for the 650,000 jobs connected with computing.

Yet we are just beginning to take advantage of the potentialities of computers. Where will we go from here? Only recently have we seen what computers are really good for and what they are not good for. Only recently have we had computers which are big enough, fast enough, complicated enough, and reliable enough to start doing many of the things that computers are well suited to do. We are at the beginning. We cannot hope to see the end. But, we can see something about tomorrow.

A computer can manipulate numbers. It can have a wide variety of applications because so many things can be represented by numbers. The direction to, distance from, and height of an airplane, missile, or satelite can be expressed in numbers. The location of an electrical component or circuit board can be specified by giving its height above and distance to the right of the lower left-hand corner of a relay rack or chassis on which components are mounted. A whole mechanical drawing or blueprint can be described by the locations of the end points of the lines and the center points and radii of circles. Flight numbers, times, code numbers for cities, and airplane capacities and numbers of reservations made are numbers which make it possible to translate airline reservation information into information a computer can handle.

A computer can compute much faster than a human being. It is versatile enough to carry out any clearly specified operation. For years, men who wanted to look into the future insisted on regarding the computer as a "giant brain" which would be made to out perform mere men. They wanted to make computers play checkers and chess, prove mathematical theorems, translate from one language into another, learn to recognize numbers, letters, words, voices and faces. Astute programmers talked freely about "artificial intelligence," and many believed seriously that the computer was about to dash beyond the human mind on all fronts.

Some small progress was made in this general direction. Dr. A. L. Samuel programmed the most powerful and expensive computer of that day, the IBM 704, to play a good but not championship game of checkers, and to improve its game by experience. Computers were taught to play poor chess. They were made to prove simple theorems of mathematics surprisingly quickly - but they could not prove complicated and useful theorems at all. While computers easily and accurately recognize the specially designed and printed numbers on checks, or can read one or several varieties of typescript, they perform badly in trying to read the very wide variety of existing type printing faces which do not bother human beings at all. Computers can read handwriting only under impractically idealized conditions. Computers can translate selected and carefully analyzed text from one language to another, but when set to translate long bodies of fresh text, they produce an ungrammatical and confusing output.

We are coming to realize that comparing a computer with human thought is like comparing an automobile with a horse or an airplane with a bird. Both man and the computer are wonderful, but they are wonderful in very different ways.

The once general enthusiasm for making computers manifest "artificial intelligence" is waning. Failure is one reason. Another is that there is really no demand for a robot chess player.

Today, a host of clever programmers and computer designers have succeeded in making computers do an increasing number of things which are useful as well as exciting. And I think it is by going further in this direction that we will accomplish many new and wonderful things in the near future.

In order to make increased usage of computers possible, there has had to be a continual process of adaptation between man and machine. At first, it was the man who had to adapt himself to the machine. For instance, each user took his program individually to the machine and used the computer either until his problem was solved, or until he ran out of assigned time.

One of the first advances in adapting computers to easy use was the "open shop" together with batch processing. Anyone who follows specified rules can get a program run. Data for a lot of short jobs which various people want done are put on a magnetic tape and run through the computer in sequence. This speeds computing immensely.

Tomorrow we will use multiprogramming. The computer will take up tasks in order of their ease or brevity. By interrupting the large job periodically, more customers will be satisfied, and no one must wait for a very long time. With multiprogramming it is possible to use in one computer many controls and many arithmetic units. This is called multiprocessing and permits several jobs to be done simultaneously. Multiprogramming and multiprocessing permit multiple access. Many users can have access to the machine simultaneously. Now, in Project MAC at the Massachusetts Institute of Technology, a number of computer users have keyboards in their offices or homes, by means of which they can call on one central computer.

Batch processing and multiple access are both steps of one particular sort toward quick and efficient use by a large number of people. Another means for adapting computers to the needs of users is simple and powerful programming languages.

Any computer is built so as to respond to a repertory of numerical instructions which cause the machine to perform arithmetical and input and output operations. These instructions, which are built into a computer, are called the MACHINE LANGUAGE of the computer. The machine language of a modern computer may consist of two or three hundred instructions or WORDS. Today, only specialists writing very special programs write in machine language. The average computer user writes in a language such as FORTRAN. FORTRAN is closely related to mathematical notation. The computer COMPILES or converts a FORTRAN program into a program in machine language.

As computers and computer programming have advanced, many new languages have become available. Among these are ALGOL (Algorithmetic Language), MAD (Michigan Algorithmetic Decoder), JOVIAL (Jules Own Version of the International Algebraic Language), COBOL (Common Business Oriented Language), COMIT (whose meaning has been forgotten, and SNOBOL (String Oriented Symbolic Language).

Some languages have special narrow purposes. ALPAK (Algebra Package) is a language for instructing the computer to add, subtract, multiply, divide, and perform other operations on algebraic polynomials. BLODI (Block Diagram) is a language useful in simulating the behavior of electrical circuits. BEFLIX (Bell Flickers) is a language useful in using the computer to make animated motion pictures.

Novice users need simple languages. Highly specialized tasks call for highly specialized languages. The languages of the future will be tailored to the various tasks which the computers of the future will be used to do.

One increasing new use of computers is handling and sorting of words and sentences. For instance, if the material ordinarily printed on library catalog cards is punched into punched cards or any other form readable by a computer, the computer can sort the entries according to classification number, and alphabetically according to author or title. The computer can then print out a listing of books in the desired order. Libraries have used a computer in this way to produce a number of copies of the entire card catalog, printed conveniently in book form.

Libraries have also used computers to prepare PERMUTATION INDEXES of various scientific publications. In a permutation index, a title is listed alphabetically under each important word in the title (excluding such words as A and THE).

Computers have also been programmed to justify (make lines of equal length) and hyphenate English test. Several American newspapers now use computers for this purpose. Further, a computer can be used to correct or alter text which has been recorded on magnetic tape, so that one can correct a manuscript without retyping it.

I believe that in the future most papers, reports, books, and business correspondence will be put into machine readable form at the first typing or keyboard operation. Such machine readable text can be sent to distant points economically over phone lines. It can serve as computer input. It can be edited without retyping. The edited copy can be printed as a book or report without a further keyboard operation. The computer will revolutionize the writing and production of reports, journals, and books.

- -

Dr. Pierce gave the above address at the 1965 MIL-E-CON Conference in Washington, D. C. It is reprinted here, with permission of the author.

High-speed digital computers are complex machines, each of which may have more than a million electrical and electronic parts. A computer programmer cannot efficiently program such a machine without first understanding how it works. This understanding should not be confined to one specific digital computer model because computer designs are continually being refined. A computer programmer, therefore, needs a general knowledge of digital computer design and operation. He must know what digital computers are, what they do, and how they do it.

The subject of digital computers covers a large number of different machines, and to discuss all of them in detail would be impractical. The information in this manual, then, is developed along general lines with the only specific information pertaining to the Burroughs D-825, Modular Processor used in the BUIC system.

HISTORY OF COMPUTERS

The word "computers" comes from the Latin verb "computare" which means to reckon or think. Thus, a digital computer is a machine that recons (or calculates) with digits.

A computing machine is actually a data processing device--that is, a device that performs mathematical and logical operations on data in a prearranged and controlled manner. To perform these operations, computing machines must be able to: (1) accept the items of data that are presented to them, (2) manipulate these items in a desired prearranged manner, and (3) make the manipulated data available in useful form.

The first devices used by man as an aid in computation were fingers, sticks, stones, and similar objects. One of the earliest "machines", the abacus, evolved from the use of pebbles as counters and has been highly touted as a computer; however, it is really nothing more than an indicating device. The abacus originated sometime before 1200 A.D. This device is one of the simplest forms of an adding or counting machine. It consists of a series of rods on which the positioning of beads records the numbers 0 through 9. Addition or subtraction can be accomplished on each bar individually. However, the carrying of the 1 when a sum is greater than 9 cannot be done automatically.

The first machine that made provisions for automatic carrying of digits, when the sum of a column is greater than 9, was the Pascal machine invented in 1642 by a French mathematician. This was perhaps the first actual accounting machine. It was used to figure currency in a customs house. Basically it was a hand-operated, gear-driven counter with addition performed by turning a wheel a distance equal to the currency to be added. In 1801, another Frenchman named Joseph Jacquard came upon the idea of punched cards. Jacquard used a chain of perforated cards to control weaving of figured fabrics on a loom. This mechanism, called the Jacquard Loom, functioned quite successfully and proved to be the basis for some remarkable developments.

The basic forerunner of the modern large-scale computers was the Babbage Analytica Engine, conceived by Charles Babbage in 1833. This machine, which operated somewhat similarly to the Jacquard Loom, made use of cards and strips of metal with various holes punched in them to record numbers. A number was represented by an equivalent number of holes. After the Babbage machine, several improved types of computing machines were developed. Notable among them was the Hollerith machine which used the Jacquard idea of holes punched in tape or cards. However, in the Hollerith machines these holes controlled electrical mechanisms.

As business and industry grew during the first part of the 20th century, the demand for accounting machines rose steadily. In 1914, a mechanical key punch, a gang-punch, a vertical sorter, and a tabulator were available to meet the accounting needs of the

nation. However, all of this equipment was electromechanical in nature, and each machine could perform only one or two basic operations. What was needed was a machine that could perform a multitude of tasks at a high rate of speed.

The first of the large-scale, high speed computing machines was the Mark I, which was completed in 1944 by Harvard University and International Business Machines (IBM). This machine uses the IBM punched-card method to insert the input data. Its output is typed out by an electric typewriter. The sequence of operations of the Mark I is controlled automatically. The machine can add, subtract, multiply, divide, or perform other related arithmetic operations. It is primarily a relay-operated device.

The Harvard Mark I was highly successful, but relay operation was undesirably slow. The first all-electronic computer was the Electronic Numerical Integrator and Calculator (ENIAC), which was built by the University of Pennsylvania in 1946. The ENIAC utilized 18,000 vacuum tubes as storage elements instead of the relays and switches used in the Mark I. It could add two 10-digit decimal numbers in 200 microseconds, or multiply them together in 2 to 3 milliseconds.

In 1951 a machine called the Universal Automatic Computer (UNIVAC) was produced by the Remington-Rand Corporation. This machine could handle alphabetic as well as numerical data. The UNIVAC proved to be the nucleus for a new computer field; namely, the large-scale, general-purpose digital machine.

One of the first uses made of a large-scale, general purpose digitial computer was to perform design calculations for the aircraft industry. In 1955, a large computer was installed at the Monsanto Chemical Company to handle mostly commercial data processing such as billing, stock inventory, payrolls, etc. However, improvements were still being made in the computer field, mainly in the type of memory to be utilized with the computer. In 1955, computers were introduced which contained magnetic cores in their high speed memory devices. One of the latest developments in the computer field has been the added capability of one computer to handle data from up to six input/output devices at one time.

The Air Force, in late 1950, enlisted the cooperation of various civilian organizations in its efforts to improve the capabilities of the United States air defense network. The overall program was known as the Continental Air Defense System (CADS) Project, which under civilian organizations helped to bring the national air defense system up to the best possible operating condition and made recommendations to ensure the system's continued effective operation. The air defense system was greatly improved by the CADS Project, but fell short of the Air Defense Command requirements for a vastly improved air defense system.

Simultaneously, studies were made on the combined use of digital computers and radar-data transmission equipment for application to air defense. The testing of a high-speed digital computer was recommended to the Air Force to provide information on the capabilities of such equipment to solve the ever-growing problem of air defense. The findings of this program led to many new concepts for solving the problem and resulted in the establishment of an experimental project which gave rise to the SAGE System. This project was developed in three major phases: the 1953 Cape Cod System, the 1954 Cape Cod System, and the experimental SAGE subsector.

The 1953 Cape Cod System was composed of a computer known as Whirlwind I (WWI) and a Direction Center, along with associated radar equipment. The purpose of this arrangement was to gather preliminary test data which would substantiate the concepts of the SAGE System then being planned. Emphasis was directed toward singling out obvious problem areas and attempting to correct whatever difficulties were encountered, rather than toward gathering complete statistical data on system operation. Consequently, there was very little modification of equipment.

The 1954 Cape Cod System was the same as the 1953 system except that radar network and mapping facilities were increased. Several minor improvements were incorporated in the operating positions within the Direction Center. The primary objective was to supply statistical results on system capacity and accuracy.

The experimental SAGE Subsector, located in Lexington, Massachussetts was completed in 1955. It was equipped with a prototype AN/FSQ-7 Combat Direction Central known as XD-1. A radar system provides a variety of inputs similar in number and type to those used in the SAGE System. An Air Force ground-to-air data link was connected to the outputs for experiments with data-link-equipped aircraft.

The experimental SAGE Subsector provides experimental data on electronic reliability, computer programs, and operating procedures. It was organized to support the regular functions of a Direction Center and to obtain operational approval and to determine required equipment modifications.

When it became apparent that there was a definite missile threat to the SAGE sites, plans were made to build hardened control centers that could survive nuclear attacks. This plan for hardened Super Combat Centers was only partly developed when in 1961 it was abandoned in favor of a less expensive back-up system, BUIC (Back-Up Interceptor Control).

The BUIC System was established to provide a capability for the conduct of air defense in the event that SAGE control capability is lost. BUIC NORAD Control Centers (or NCCs) have a higher probability of surviving a missile attack than SAGE DCs because they are co-located with selected long-range radar (LRR) sites that are not near expected ICBM targets. The BUIC System consists of three phases: BUIC I, BUIC II, and BUIC III. BUIC I was made up of 27 MANUAL control centers which provided immediate back-up capability similar to that provided by the old Manual Air Defense system. Some BUIC I NCCs will remain in less critical areas.

BUIC II consists of 13 computerized NCCs, each capable of taking over the air defense responsibilities of a SAGE DC. BUIC NCCs do not have the same capability as SAGE DCs, but they are capable of performing similar functions in a similar, computer assisted manner.

The BUIC II system is being replaced by the BUIC III system which provides improved and expanded capabilities.

The computer equipment used in the BUIC III system is a military version of the Burroughs D-825 Modular Processor. The basic computer equipment is designated AN/GYK-10 and the full BUIC configuration, including the AN/GYK-10, is called AN/GSA-51A.

The Burroughs military oriented D-825 was conceived, designed and developed for one specific objective: to fulfill the computational requirements of present and future military and space applications. It comes under the category of second generation computers which means that it relies on solid state circuitry (first generation computers use vacuum tubes). A typical example of the growing complexity of tasks computers must perform in military applications is the intricate and diverse problems encountered in a command and control operation. The D-825--with its tailor-made module configuration, programming ease and versatility, parallel processing, automatic diagnostic routines, high-speed thin film "scratch pad" memory, and totally shared main memory modules--fulfills all requirements.

In 1964 IBM announced that it had taken a "billion dollar gamble", and introduced the IBM System/360. It had been thought that IBM was planning to produce some new equipment, but few dreamed they would introduce machines that would replace all of their previous computer lines. Large configurations of the System/360 have a primary storage capacity of 8 million characters, and this can be expanded by external storage devices. The System/360 is truly a third generation computer, with its design based on microelectronic hybrid integrated circuits.

Where will we go from here? Well, in the past decade the speed of computation has increased a million times, however, we are rapidly reaching the limits (according to Einstein's theories) as far as speed is concerned. In the future then, computers may not be doing things so much faster, but will be operating in many new applications and fields. Computers have already designed the circuitry for new computers. As a programmer you can expect the computer to do more of the labor by producing more powerful languages. There should be an increase in the capacity of memories with a reduction in the physical size of computing equipment.

BASIC CLASSES OF COMPUTING MACHINES

DIGITAL COMPUTERS

A digital computer is a computing machine that processes data expressed as digits or numbers, manipulates the data by means of arithmetic or logical control operations in a predetermined manner, and generally delivers the resulting information in the form of digits. As an example, the number 34 might be represented in a digital computer as shown in Figure 1.

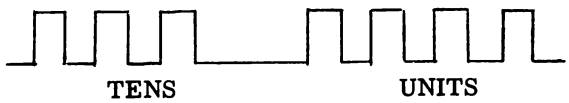


FIGURE 1
Digital Form

A digital computer operates on data in much the same way that a man would manipulate the data in carrying out arithmetic computations with pencil and paper. Similar to a man making an arithmetic computation, a digital computer sequentially manipulates digits.

The digits used to represent either items of data or specific instructions for processing the data must belong to a particular number system (such as the familiar decimal system) chosen for the computer model being used. Similarly, the results of operations by a digital computer are usually delivered in the form of numbers.

Basically, digital computers have the following characteristics:

(1) All data handled by the computer must be in the form of digits of a particular number system.

(2) The computer processes data by performing predetermined arithmetic and logical control operations on the digits. These operations are performed in discrete steps, much as arithmetic operations are performed with pencil and paper.

ANALOG COMPUTERS

An analog computer, unlike a digital computer, is a computing machine in which data is converted, for purposes of computation, not into digits, but into physically measurable quantities such as lengths, angles, or voltages (as shown in Figure 2). Computed results are obtained by the action of moving parts or electrical signals. These actions or signals do not represent digits. Rather, they are related to one another in such a way as to represent the relationships among the terms of a mathematical equation. They also interact with one another in such a way as to represent the mathematical operations indicated in the equation.

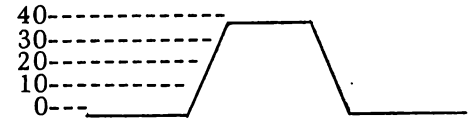


FIGURE 2
Analog Form

In other words, an analog computer solves problems by causing physical quantities to vary in a manner analogous to the way in which the variables in a problem change. For example, if distance equals velocity multiplied by time, a motor running at a speed proportional to velocity during a given time interval will turn a gear train through an angle proportional to distance. Thus, a continuous solution of distance in the equation, Distance = Velocity X Time, may be obtained. Action of this kind is typical of the manner in which analog computers solve problems. A fundamental characteristic of analog computers is that they provide continuous solutions to a given problem.

DIGITAL COMPUTER ELEMENTS

As explained previously, a digital computer works with digits or numbers. In order for a digital computer to operate, certain elements are required for the proper handling and manipulation of data, just as a man needs certain tools to perform arithmetic tasks. This comparison is easily supported by describing the elements of the computer that correspond to a man working at a desk. Let's assume that the man is a clerk working in a payroll office and is computing the net pay of various individuals. The "IN" box on his desk contains the pay rates of the personnel involved plus miscellaneous data such as the initiation of bond deductions, etc. A digital computer has an input element which is capable of accepting various types of data and presenting it to the computing portion of the equipment. The clerk has several tables to which he refers, such as tax deduction tables, standard weekly deductions applicable to each employee. In a digital computer, the memory element would serve as the temporary storage device for all these facts. The actual computation of an individual's salary is done in the payroll clerk's head, or perhaps with a desk calculator; in either case, this function is the same as that performed by the arithmetic element of a digital computer. Once the net pay of each person has been calculated, the clerk fills out a standard form which contains the employee's name and the amount due. He then places all their forms in the "out" box on his desk, thus completing his job. The output element of a digital computer accepts the results of computation by the arithmetic element and presents the results in a form recognizable by the user. Of course, all the actions of the payroll clerk are controlled and coordinated by his nervous system. The control element coordinates the actions of a digital computer and is connected to all the other elements. From this discussion, we can see that a digital computer is essentially composed of the following elements:

(1) Input

(2) Output

(3) Memory

(4) Arithmetic

(5) Control

The operational elements listed above are the elements required by typical digital computers. The following paragraphs describe these elements in more detail and explain the tasks performed by each. Examine Figure 7.

INPUT ELEMENT

The input element is capable of accepting data in a variety of forms and converting it to a standard format which the rest of the computer elements can use. For example, in Figure 3, a typewriter input might be used; accordingly, the operator would type out the data and instructions in a decimal code. The typewriter would have switches connected to each key which would convert the hitting of a key into an electrical impulse. The electrical impulse might then be converted to a binary code so that the computer could work with it. The type of inputs which constitute the input element for different machines varies a great deal; therefore, it is not possible to say that any one combination of units make an input element. Other common types of input devices are punched card readers, magnetic tape readers, paper tape readers, and such automatic input units as telephone lines which transmit data from remote locations. The input element provides one-way communication between the external sources and the computing elements. Data and instructions are fed to a computer through an input element, but an input element returns nothing to the external sources.

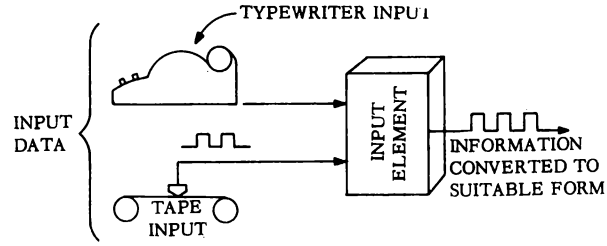


FIGURE 3
Input Element - Receives Information and Converts It Into Usable Form

OUTPUT ELEMENT

The results of a digital computer's operations must be delivered to the user of the machine in an appropriate form. The element that accomplishes this transfer is the output element. The results of a computer's operations, however, are not necessarily in the form best suited for use outside the machine. Hence, an output element may include facilities for converting the results of the computer's operations into the form of output data best suited to the user of the machine. For example, in Figure 4, the answer to the problem might enter the output element in the form of binary electrical pulses. The output element may then convert these pulses to voltages that operate either an electrically operated typewriter or a printing machine to print the final answer.

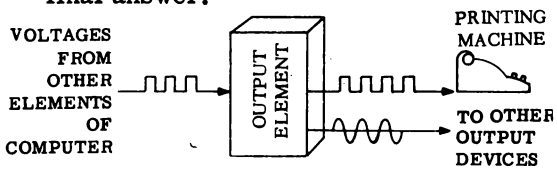


FIGURE 4

Output Element - Converts Computer's Answers Into Form Usable By External Output Devices

Like the input element, the output element makeup is flexible and varies widely from one computer to another. Some common types of output devices are line printers, card punches, magnetic tapes, and visual indicators.

The output element, like the input element, is a one-way unit. It receives information from the other elements of the computer and transfers the information to the final user, but it does not return any information to the computer.

ARITHMETIC ELEMENT

Since the purpose of a digital computer requires that the machine perform arithmetic operations on the input data, a digital computer must obviously contain an element that can accomplish these operations. This is the arithmetic element. All data to be operated on arithmetically must enter this part of the computer. Likewise, most instructions determining what computations are to be performed must control the arithmetic element. (See Figure 5.)

Theoretically, it would be possible to build an arithmetic element which could perform most mathematical operations directly, just as a man performs them. This, however, would require a very large and complicated arithmetic device; consequently, it is never done. Instead, the arithmetic element is usually designed to perform only a few basic operations such as addition, subtraction, multiplication, and division. (Subtraction, multiplication, and division are simple variations of the addition function.) If an arithmetic device can perform either addition or subtraction and a few other simple operations, it can be made to perform almost any other mathematical operation by simply breaking the operation down into its fundamental operations. This is the way in which the arithmetic element is made to do the more complex mathematical operations that are often required.

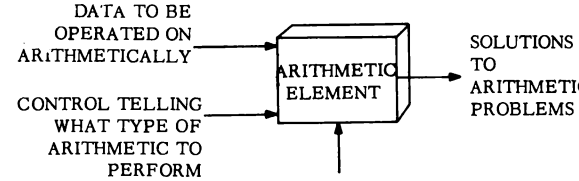


FIGURE 5

Arithmetic Element - Data Enters and Is Processed by This Element

MEMORY ELEMENT

As explained previously, operations in a digital computer are carried out in step-by-step fashion. For this reason, some of the information fed into a computer must be stored for indefinite periods prior to actual usage. The facilities required for storing information in a computer are included in the memory (sometimes called main storage) element.

Information fed into a computer includes:

(1) Particular items of data to be processed.

(2) Instructions (known as the program) for performing the particular data processing operations required.

(3) Reference data.

The memory element comprises a large number of storage locations in which information can be stored until it is needed by one of the other elements. Each of these locations has an absolute address assigned so that it may be selected by the computer for insertion or extraction of data. For instance, a typical computer instruction might be to "add the quantity which is stored in location 1000". The instruction which stated that address 1000 contained the desired OPERAND is also stored in the memory element; this makes the memory element a "shared" device.

Many types of storage devices such as magnetic cores, magnetic tapes, magnetic drums, acoustic delay lines and cathode-ray tubes are used in memory elements. At present, magnetic cores are the most popular device, primarily, because of their high speed and stability.

CONTROL ELEMENT

There must be a definite sequence for the flow of data during processing by a digital computer. For example:

(1) Data must be inserted into particular storage locations and then used in correct sequence at the appropriate times.

(2) The arithmetic element must also be "told" what operations to perform on the data and in what order to perform them.

(3) The results of the arithmetic operations must be routed to the appropriate storage or output locations.

(4) The transfer of all output data to the output element and to the ultimate user must be properly controlled to ensure the required sequence of information.

The entire sequence of operations by the computer is predetermined by the program (and the construction of the computer) for the data-processing task. The program, coded in the digital language, is inserted through the input element and stored at specific addresses in the memory element. The element for interpreting and carrying out instructions contained in the program is the control element. (See Figure 6.)

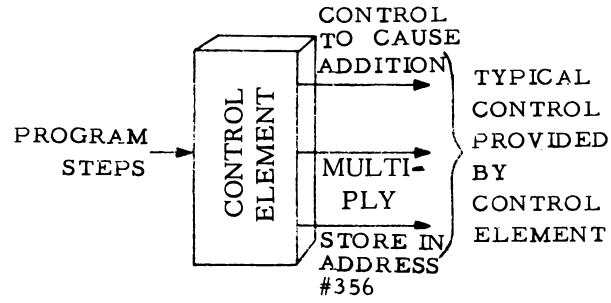


FIGURE 6
Typical Instructions From the
Control Element

The instructions are transferred from memory to the control element, where they are decoded, and certain commands are set up by the control element. One of these commands is to go into memory again and transfer out the designated operand to the arithmetic element. Following the calculation in the arithmetic element, the results of a desired operation are usually programmed to be returned to memory and then transferred to the output element during an alloted time interval.

By its interpretation of the program, the control element governs the flow of data and the sequence of operations performed by the computer, special electrical circuits provide the required control. These circuits respond to electrical signals representing the digits that make up the control instructions to program and produce appropriate control signals. The control signals cause arithmetic operations to take place and effect the transfer of data from one element of the machine to another. For example, in Figure 7 the action of the control element on the other elements is shown. In this figure, information transfer is shown in heavy lines while control lines are light.

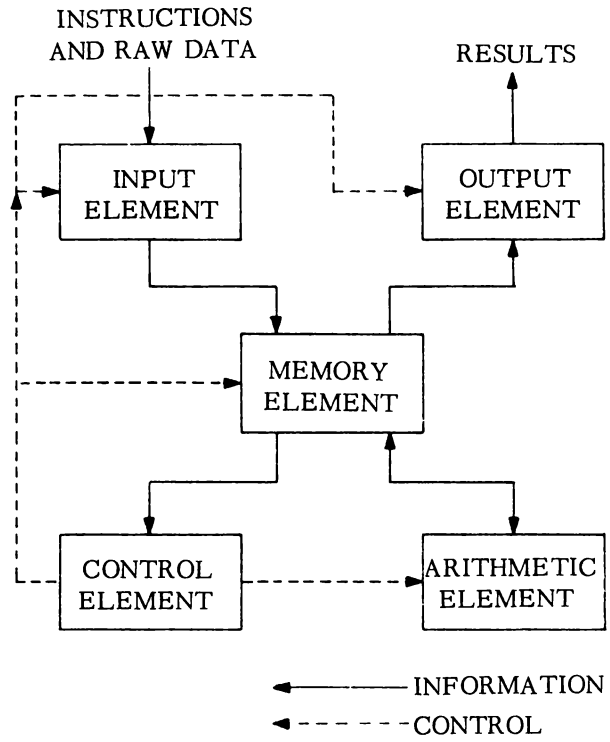


FIGURE 7
Digital Computer, Block Diagram

SUMMARY

Figure 7 shows the five elements we have been discussing arranged as a typical computer system. Information, which includes computer instructions as well as data, enters the computer through the input element where it may be converted to a common form and placed in a buffer storage device. Next, the memory element accepts this information at specific intervals and places it in the proper storage location. Notice that there are two paths leading out of the memory element--one to the control element and the other to the arithmetic element. Earlier, we spoke of the memory element as a shared device, this is true because it serves as the memory for two other elements.

NUMBER SYSTEMS

Prior to studying the programming of a digital computer it is necessary to learn something about the language of the computer. This is similar to an electronics man first understanding the volt, ohm, and ampere before attempting to master the trouble-shooting of a receiver or transmitter. The computer language is basically very simple; and once the language is mastered, the fundamentals of programming will be more meaningful. Most computers understand and speak in the language of binary mathematics; however, the preparation of information for a digital computer involves the use of the decimal, octal, and binary number systems. Since a computer processes information that is expressed or coded in numerical form, a brief investigation of some available number systems is in order.

In general, a number system has three basic uses:

(1) It is a method of counting in order to be able to express a quantity. For example, the tally for a count of 13 is 𝍸 𝍸 III.

(2) It is a method of arranging symbols in a specific sequence in order that one's relative position among others may be known. For example, the address on a home indicates the position of the house with respect to a given street and other houses.

(3) It is used to provide symbols for unique identification. For example, military serial numbers and social security numbers are used as a means of positive identification.

There is no stipulation that the symbols (characters) used in a numbering scheme be the familiar Arabic numerals. Roman numerals, for example, are used extensively to indicate chapters in textbooks. Moreover, there is no rule that compels the use of decimal notation. We are all familiar with one numbering system that is not truly decimal--our own military time system. In this unique system, the unit of measurement changes at quantities other than 10; it is easy, nevertheless, to see how a carryover into the next higher significant position is accomplished. For example, 1750 (5:50 PM) plus 20 minutes equals 1810 (6:10 PM).

The familiar decimal system is the most universally known numbering system. It derives its name from the total number of symbols used--decim is the Latin word for ten. The decimal system uses as symbols the ten Arabic numerals--0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. These symbols are used alone or in combinations to express quantities or identities. The ten symbols have a commonly known order or sequence which enables us to count and to indicate quantity with them. The radix, or base, of the decimal system is ten. The base of a counting system is determined by the number of different symbols used; furthermore, the radix determines the number of counts possible in one counting cycle. Using the decimal system, how many counts can you make before a carry out to the next significant position occurs? Obviously, a single decimal number can indicate any count or value from zero through nine, inclusive; therefore, we can count up to, but not including, 10.

There are many nunbering systems in common use today. A number of these systems are not numerical in nature. An example of one of these is the Roman Numeral System. The disadvantage of systems such as this is the difficulty encountered in performing the basic operations of multiplication, division, etc. Our decimal system is very handy for us to use and it seems like the "natural" system. It is no more "natural" than any other system. In fact, the reason we used it was probably because we first counted on our ten fingers. Unfortunately, the digital computer is not adapted to the decimal system. As a result, we will need to become familiar with other numbering systems.

There are two terms common to all numbering systems. An understanding of these terms will lead to an understanding of the other systems.

The first term is RADIX. The radix is the BASE of a numbering system. In the decimal system the radix is 10, octal system - 8, quinary system - 5, and binary system - 2. Each of the numbers shown in Figure 8 represents an equal quantity. The only difference is in the radix of the system in which the quantity is expressed.

RADIX
$165_{(10)}=245_{(8)}=1130_{(5)}=20010_{(3)}=10100101_{(2)}$

FIGURE 8
A Quantity Expressed in Several Numbering Systems

The radix limits the stock of numbers available in any numbering system. In the decimal system with a radix of 10, there are only 10 different distinct digits--0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. When we exceed 9, a combination of these same basic digits must be used.

In the octal system with a radix of 8, only 8 different distinct digits exist--0, 1, 2, 3, 4, 5, 6, and 7. There is no 8 or 9 in this system. When we exceed 7, we must use again a combination of the basic digits 0 through 7.

In the binary system, radix 2, there are only two distinct digits--0 and 1. There is no 2, 3, 4, etc., in this system. To express numbers greater than 1, combination of 0 and 1 must be used.

The idea contained in the last three paragraphs is illustrated in figure 9. Another useful point to note from this figure is that the largest digit in any system is always one less than the radix.

The second term common to all numbering systems is PLACE VALUE. Place value is the weight of a digit. The place value is determined by the position of the digit relative to the point of the system. The point is called decimal point, octal point, or binary point, depending upon which system the number is expressed. Place value is the only difference between the numbers shown in Figure 10. Zero has no value. It is used only to determine the place value of the other digits.

DECIMAL	OCTAL	BINARY
0	0	0
1	1	1*
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7*	111
8	10	1000
9*	11	1001
10	12	1010
11	13	1011
12	14	1100
*LARGEST DISTINCT DIGIT		

FIGURE 9
Distinct Digits of Decimal, Octal and Binary Numbering Systems

495	400095
4905	490005
4095	409050
409005	495000

FIGURE 10
Illustration of PLACE VALUE

To determine the place values of a numbering system, the radix is raised to ascending powers beginning with zero and moving to the left from the point. The place values of the decimal system would thus be determined as shown in Figure 11. The actual place values are found by performing the indicated operations. Note that any number raised to the zero power is mathematically defined as 1. From this figure it is seen

that each place value is 10 times the preceding place value.

10^5	10^4	10^3	10^2	10^1	10^0	.	DECIMAL POINT
100000	10000	1000	100	10	1	.	PLACE VALUES

FIGURE 11
Place Values of Decimal System

The decimal number 365 is shown in Figure 12. This example shows that when each digit of the number is multiplied by its respective place value and the products are added together, the result is the value of the number expressed in the decimal system, in this case, 365.

10^5	10^4	10^3	10^2	10^1	10^0	.
100000	10000	1000	100	10	1	
			x3	x6	x5	
			300	+ 60	+ 5	= 365

FIGURE 12
An Example of a Number
Expressed in the Decimal System

Here is a general equation that applies to any number expressed in any number system:

$$V_R = D_1R^0 + D_2R^1 + D_3R^2 + - - - D_NR^{N-1}$$

Where V_R = Decimal value of the number.

D = Digits of the number.

R = Radix of the system in which the number is expressed.

N = Number of digits in the expression.

By substitution into the equation, the decimal equivalent of any number in any system may be determined. This is the normal procedure used for converting to decimal.

The octal numbering system, base 8, is used extensively in the field of computers. The place values of this system are shown in Figure 13. Notice the lowest place value is 1 since any number to the zero power is 1. Also, note each place value is 8 times the preceding place value.

8^5	8^4	8^3	8^2	8^1	8^0	.	OCTAL POINT
32768	4096	512	64	8	1		PLACE VALUES

FIGURE 13
Place Values of Octal System.

Consider an octal number such as $3720_{(8)}$. When each digit of this number is multiplied by its proper place value and added, as shown in Figure 14, the result is $2000_{(10)}$. This means that $3720_{(8)} = 2000_{(10)}$.

8^4	8^3	8^2	8^1	8^0
4096	512	64	8	1
	x3	x7	x2	x0
	1536 +	488 +	16 +	0 = 2000

FIGURE 14
An Example of a Number Expressed in the Octal System

Perhaps it seems strange to think of a number expressed in some system other than the decimal. However, this is not a new concept. Other numbering systems are commonly used everyday. The English system of inches and feet has a base of 12. The time measuring system of seconds, minutes and hours has a base of 60. The addition of two numbers in the latter system is shown in Figure 15. Notice when the base, 60, is exceeded, a carry is generated and added into the next column to the left. The amount by which the base is exceeded is the sum for that column. This same procedure holds for addition in any numbering system.

```
  1 -- CARRY
 10: 40
 10: 45
 11: 25
```

FIGURE 15
Addition of Two Numbers in a System, Base 60

The majority of the circuits in computers are two-state devices which can be placed in either one of the two states. In computer logic, we call these two states "YES" and "NO", "ON" and "OFF" or "HIGH" and "LOW". This means that the numbers that these two-state devices can represent must have no more than two different distinct digits, 0 and 1. We can, therefore, just as well call these two states 0 and 1. This is the binary numbering system, base 2.

The place values of the binary numbering system are shown in Figure 16. The lowest place value must be 1. Each place value is two times the preceding place value.

2^6	2^5	2^4	2^3	2^2	2^1	2^0	.	BINARY POINT
64	32	16	8	4	2	1		PLACE VALUES

FIGURE 16
Place Values of Binary System

Any decimal number may be expressed by some combination of ones and zeros in the binary system. Figure 17 shows the decimal numbers 0 - 9 and the corresponding binary number. Notice that decimal value of any binary number may be determined by adding the place values of the columns containing 1's.

2^3	2^2	2^1	2^0	.	
8	4	2	1	-	Place Values
			0		0
			1		1
		1	0		2
		1	1		3
	1	0	0		4
	1	0	1		5
	1	1	0		6
	1	1	1		7
1	0	0	0		8
1	0	0	1		9
BINARY					DECIMAL

FIGURE 17
Decimal - Binary Conversions

As a comparison between the decimal and binary systems, consider the number $43_{(10)}$. Figures 18 and 19 show this number in both systems. Notice in each system that the digits multiplied by their respective place values add together to produce $43_{(10)}$.

1000	100	10	1	PLACE VALUES
		x4	x3	DECIMAL NUMBER
		40	+ 3 = 43	

FIGURE 18
Decimal System

64	32	16	8	4	2	1 .	PLACE VALUES
1	0	1	0	1	1		
32	+	8	+	1	+	1 = 43	BINARY NUMBER

FIGURE 19
Binary System

The binary number 101011 in Figure 18, which is equal to $43_{(10)}$ may be found in the following manner. A "1" is placed in the sixth column because 32 is the largest place value not exceeding 43. This expresses 32 of the original 43. Eleven remains to be expressed. Sixteen is too large, so a "0" is placed in the fifth column. A "1" in the fourth column expresses 8 of the 11. The remainder to be expressed is now 3. Four is larger than 3, so a "0" is placed in the third column. A "1" in the second column expresses 2 units of the quantity 3. The remainder is now "1" which may be expressed by placing a "1" in the first column. This is one method of arriving at the number 101011. A shorter, more direct method of converting from decimal to binary exists and will be shown later.

NUMBERING SYSTEM CONVERSIONS

There is a problem associated with binary numbers. They become extremely difficult to handle because of the number of bits (contraction of Binary digIT) required to represent large numbers. To avoid this disadvantage, the octal numbering system is used to express binary numbers. The octal system is used because it is extremely easy

to convert from binary to octal and vice versa.

To convert from binary to octal, the binary number is set off in groups of three bits beginning at the binary point as shown in Figure 20. Each group of three bits is then converted directly into octal. The procedure for converting from octal back to binary follows the same rules in reverse.

1 0 1	1 0 0	0 1 1	1 1 1 .	BINARY NUMBER
5	4	3	7 .	OCTAL NUMBER

FIGURE 20
Conversion Between Binary and Octal Systems

Almost all numbers encountered outside computers and in our environment are decimal. This means that to place them in a computer they must be converted to binary. There are two methods of doing this. One method is to go directly from decimal to binary. The other method is to convert from decimal to octal and then by inspection, convert from octal to binary. These two methods are shown in Figure 21 (a), (b), and (c).

The procedure shown in Figure 21 is called successive division. The decimal number is divided by the base of the new numbering system, 2 and 8 in this case, successively. The remainders from each division are saved and form the digits of the new number.

This discussion of numbering systems is basic to the understanding of the mathematics of digital computers.

Now, in order to convert octal integers to their decimal equivalents the process of successive multiplication is used. In this process the left most digit (also called the Most Significant Digit or MSD) of the octal value is multiplied by its radix. To this product is added the next digit to the right. The sum is multiplied by the radix, and again the next digit is added. Addition of the last digit is the final step; do not multiply again at this point. (See Figure 22.)

(a) Converting $647_{(10)}$ to binary

```
2 |647          REMAINDERS
  2 |323 - - - - - - - 1 <- LOWEST PLACE
    2 |161 - - - - - - 1   VALUE COLUMN
      2 |80 - - - - - 1
        2 |40 - - - - 0
          2 |20 - - - 0
            2 |10 - - 0
              2 |5 - - 0
                2 |2 - - 1
                  2 |1 - - 0
                    0 - - - 1 <- HIGHEST PLACE
                                 VALUE COLUMN
```

$647 = 1010000111$

(b) Converting from decimal to octal

```
8 |647          REMAINDERS
  8 |80 - - - - - - - 7 <- LOWEST PLACE
    8 |10 - - - - - - 0   VALUE COLUMN
      8 |1 - - - - - - 2
        0 - - - - - - - 1 <- HIGHEST PLACE
                             VALUE COLUMN
```

$647_{(10)} = 1207_{(8)}$

(c) Converting from octal to binary

1	2	0	7	OCTAL
001	010	000	111	BINARY

FIGURE 21
Conversion From Decimal to Binary

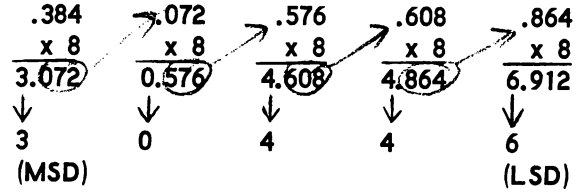
Example: $135_{(8)}$ is equivalent to. . . . .(10)

```
  1  3  5
 x8   |
 --   |
  8   |
 +3 <-|
 --
 11
 x8
 --
 88
 +5 <---
 --
 93
```

$135_{(8)} = 93_{(10)}$

FIGURE 22
Conversion From Octal to Decimal

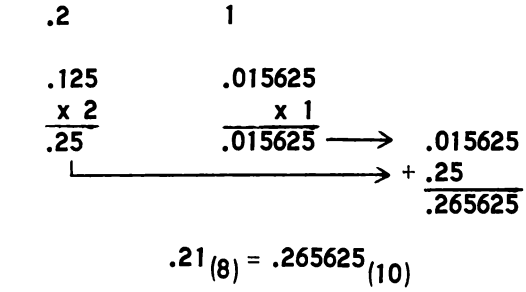
The fractional portion of a decimal number may be converted to its octal equivalent by employing the multiplication method. To convert the fractional portion of a decimal number to an octal fraction, multiply the decimal fraction by eight. The integral portion of the product thus obtained is the MSD of the octal fraction. The fractional portion of the product should be multiplied by eight. This time, the integral part of the product is the second digit of the octal fraction. Repeat the process until the fraction of the product is zero, or until sufficient octal digits have been generated. Study the following conversion of $.384_{(10)}$ to its octal equivalent:



The results of the conversion show that $.384_{(10)}$ is equivalent to .30446.....$_{(8)}$.

Of all the methods of conversion of fractions from octal to decimal the direct or expansion method is about as simple as any. This consists of multiplying the octal digits by their place values, and then adding these values together. The place value for octal fractions is derived from the negative powers of eight. In the following example we have the octal fraction .21 and we want to find its decimal value.

NEGATIVE POWERS OF EIGHT			
8^{-1}	8^{-2}	8^{-3}	8^{-4}
0.125	0.015625	0.001953125	0.000244140625



$$.21_{(8)} = .265625_{(10)}$$

ARITHMETIC OPERATIONS

Thus far you have become acquainted with some of the characteristics of numbers and number systems, and you have learned how to convert numbers from one system to another. Now, it is time for you to apply your knowledge of the number systems--you are now ready to perform arithmetic operations in both the octal and the binary systems.

BINARY ADDITION

The addition of two numbers in the binary system is extremely simple. You are familiar with the decimal arithmetic performed by a counter, which gives the proper sum and produces a carry when it counts from "9" to "0". Such a counter must be capable of handling 100 possible combinations of inputs--each of the two numbers to be added may be any one of the digits from zero through nine. A binary adder, however, has only four different combinations to cope with--0 + 0, 0 + 1, 1 + 0, and 1 + 1. If both inputs are "0's", the counter produces a sum of "0" and no carry. If either input contains a "1", the adder produces a sum of "1" and no carry. If both inputs are "1's", the adder produces a sum of "0" and a carry into the next column. Therefore, the simple rules for binary addition are as follows:

0 + 0 = 0

1 + 0 = 1

0 + 1 = 1

1 + 1 = 0 and 1 to carry

These rules apply in all cases of addition; furthermore, they apply to the addition of both integers and fractions. Binary numbers, like decimal numbers, are added from right to left, and the carry is added to the adjacent column on the left.

The technical terms in addition are defined as the augend, addend, and the sum. The augend is the term that is to be increased; the addend is the term to be added to the

augend; and the sum is the result of the operation. For example:

```
 1101   AUGEND
 1011   ADDEND
11000   SUM
```

The addition of more than two numbers is handled thus: The addition of the first set of numbers is performed, then the third number is added to the original sum. To the sum of each succeeding addition, add the next number until all numbers have been totaled. For example, add:

```
  011
  111
  110
 +101
 ----
 1010
```

ADDITION OF THE FIRST SET OF NUMBERS	011
	+111
FIRST SUM	1010
ADDITION OF THE THIRD NUMBER	110
SECOND SUM	10000
ADDITION OF THE FOURTH NUMBER	101
FINAL SUM	10101

Binary fractions are added in accordance with the rules that govern whole numbers. The binary point is fixed, just as it is in the decimal system. Study the following two additions:

a. DECIMAL	BINARY
.125	.001
+.375	+.011
.500	.100

b. DECIMAL	BINARY
.5	.1
+.5	+.1
1.0	1.0

A perfectly legitimate operation within a computer's arithmetic element may result in a number beyond the computer's capacity. The result of an addition, for example, may be outside the computer's range. This is analogous to increasing by "1" an automobile odomter that already indicates a maximum of 99,999 miles. Increasing the count by one cannot possibly cause the odometer to indicate 100,000 miles. Accordingly, the result of an addition may be too great for a computer to handle. Adding two positive numbers should give a positive result, and adding two negative numbers should give a negative result. If the sum is too large for the computer word, then the answer is incorrect. This is called overflow, and in BUIC an indicator bit called the program overflow bit (POV) is set.

BINARY SUBTRACTION

Subtraction is addition in reverse. Like addition, it is basically a process of counting, but in the opposite direction. For example, 6 - 5 = 1. (Count six, change direction and count five more, thereby you arrive at a count of one.) It is difficult to construct an electronic counter that can count backwards--counters are not electronically ambidextrious; they count in only one direction. It is possible, however, to construct a machine that will subtract by adding complements. In other words, subtraction in a computer is the same as addition, except that the complement rather than the number itself is added.

The technical terms in subtraction are defined as the minuend, subtrahend, and the difference. The minuend is the number to be decreased; the subtrahend is the quantity of the decrease; and the difference is the result of the operation. For example:

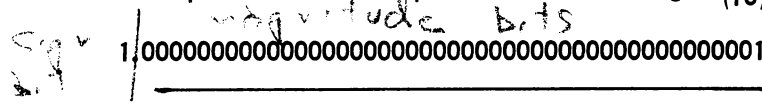
```
 111   MINUEND
-000   SUBTRAHEND
 111   DIFFERENCE
```

Computer design requirements do not allow for borrowing; consequently, the complement method of subtraction fits in with computer design capabilities. The "1's" complement in binary arithmetic is nothing more than the original number with the bits reversed--that is, the original zeros are ones in the complement, and the ones are zeros. For example, the complement of 100 101 110 is 011 010 001.

ORIGINAL NUMBER	100 101 110
COMPLEMENT	011 010 001

In order to designate the sign of a value, one bit position is set aside from the rest of the bit positions in a computer word. This "sign" bit is usually the first bit position and is written with a point after it. The bit positions following the sign bit and point are referred to as magnitude bits. The point is called the machine point since it should not be confused with a decimal point or binary point.

Example: A D-825 computer word containing $-5_{(10)}$:



There are many methods of using this concept of complementing in different computers so we will be concerned only with the methods used in the D-825. The method discussed so far is called the "radix minus one" complement where we reverse the bits. Since the radix minus one in binary is a one this is also called the one's complement. Actually, when we reverse the bits we are subtracting the value from its modulus minus one.

The modulus of a number is a positive value that expresses the limit of the value according to its place value. For example, the modulus of the binary value 111 is 1000. Examine the following values and their moduli:

VALUE	MODULUS
$1011_{(2)}$	$10000_{(2)}$
$361_{(10)}$	$1000_{(10)}$
$5_{(10)}$	$10_{(10)}$
$9999_{(10)}$	$10000_{(10)}$

Now, to find the "radix" complement of a value we simply subtract it from its modulus:

VALUE	ITS RADIX COMPLEMENT
$1011_{(2)}$	$0101_{(2)}$
$361_{(10)}$	$639_{(10)}$
$5_{(10)}$	$5_{(10)}$
$9999_{(10)}$	$0001_{(10)}$
$1011100_{(2)}$	$0100100_{(2)}$
$1111_{(2)}$	$0001_{(2)}$

As you see, with the radix complement (or 2's complement) the complement of a value is not simply a reversal of the bits.

To find the radix minus one complement of a value you simply subtract it from its modulus minus one. If the binary value contains 4 places as 1101, then the modulus is 10000, and the modulus minus one is 1111. Now, to find the radix minus one complement of 1101 you subtract it from 1111.

Example:	1111	modulus minus one
	-1101	value
	0010	complement of the value

But how can the computer subtract to find the complement this way? It can't, but as you can see the radix minus one complement is the same as if we had reversed all the bits. This is a simple electronic task for the computer.

The instruction for Binary ADdition in the D-825 is BAD. When the computer encounters this instruction it will add the two values specified. If one of the values is a negative number then one of the values will be complemented according to the radix minus one. This is a form of subtraction since the answer will be the difference.

Since the complement of a value may look much larger than the value itself there is a possibility of a carry when there is actually room for the answer in the computer word. This is called an end carry (or end around carry) and is added to the least significant position of the value.

EXAMPLE:

a. End carry, no overflow:

```
 5          0.101                  0.101
-2    or    1.010    complement    0.101    add
                                   | 010
                                   └→1          or    +5
                                   -----              -2
                                   0.011              +3
```

b. End carry, no overflow:

```
 7
-5    or    0.111                  0.111
             1.101    complement    0.010    add
                                   | 001
                                   └→+1         or    +7
                                   -----              -5
                                   0.010              +2
```

c. Overflow, POV is set, answer is not correct:

```
-5    or    1.101                  1.101
-3          1.011    complement    0.100    add
                                   | 001
                                   └→+1         or    -5
                                   -----              -3
                                   1.010              -8    not   -2
```

Predicting overflow is easy with small values, but how about the D-825 computer word with its sign bit followed by 47 magnitude bits? Sometimes the end carry indicates overflow, but other times it does not. Therefore, we depend on a set of rules for adding (BAD) and subtracting (BSU) in order to find what our answer means. Figure 23 may be used to solve problems as the computer would. Remember, when complementing a value complement all bits including all leading zeros, trailing zeros, and the sign bit. When adding, add all magnitude bits, but do not add the sign bits.

BINARY MULTIPLICATION

Everyone is accustomed to performing multiplication by employing a memorized multiplication table. Given the problem 4x3, we recall the product, 12. The multiplication table is actually a short cut method of addition. In other words, "4 x 3" is the same as "add 4 three times." Multiplication is nothing more than repetative additions. A computing machine normally performs multiplication in this crude manner.

Nevertheless, a programmer should become acquainted with the binary multiplication table:

0 x 0 = 0

1 x 0 = 0

0 x 1 = 0

1 x 1 = 1

The technical terms in binary multiplication are the same as those in decimal multiplication. The multiplicand is the number to be multiplied, the other number is called the multiplier, and the product is the result of the operation. Thus:

```
1000    Multiplicand
x  1    Multiplier
----
1000    Product
```

BAD, binary add:

1. **Like signs . . . add and keep the sign.**
   a. **End carry indicates POV set.**
   b. **No end carry indicates correct answer.**
2. **Unlike signs . . . complement the augend, add and keep the sign.**
   a. **End carry indicates correct answer.**
   b. **No end carry indicates the need to complement for the correct answer.**

BSU, binary subtract:

1. **Like signs . . . complement the subtrahend, add and keep the sign of the minuend.**
   a. **End carry indicates correct answer.**
   b. **No end carry indicates the need to complement for the correct answer.**
2. **Unlike signs . . . add and keep the sign of the minuend.**
   a. **End carry indicates POV set.**
   b. **No end carry indicates correct answer.**

FIGURE 23
Rules for D-825 BAD and BSU Instructions

The general procedure when mutliplying two binary numbers is the same as that used in decimal arithmetic. The procedure is illustrated below:

Multiplicand	1001	
Multiplier	1011	
	1001	First Partial Product
	1001	Second Partial Product
	0000	Third Partial Product
	1001	Fourth Partial Product
Total Product	1100011	

Notice that the product of the two 4-bit numbers is seven bits long; however, the multiplication of the two largest 4-bit numbers (1111 x 1111) results in a product that is eight bits long. In other words, the largest product that can result from multiplication of two binary numbers will not be longer than the sum of the bits in the multiplier and multiplicand.

BINARY DIVISION

Division is the antithesis of multiplication. Computer multiplication, you recall, is a series of additions; conversely, computer division is a series of subtractions. In other words, binary division is the process of counting the number of times that the divisor can be subtracted from the dividend before a negative remainder results.

Division is often not used in a computer because the circuitry for it is relatively complex: however, the division of "y" by "x" can be avoided by simply multiplying "y" by the reciprocal of "x".

Direct division of binary numbers is accomplished according to the same rules for division of decimal numbers. The process of division is particularly simple in the binary system because the divisor, dividend, and

quotient are composed entirely of "1's" and "0's". Study the following example of $11\ 111\ 000_{(2)}$ divided by $1000_{(2)}$:

```
                  11111  <-- quotient
divisor --> 1000 /11111000 <-- dividend
                 1000
                 ----
                  1111
                  1000
                  ----
                   1110
                   1000
                   ----
                    1000
                    1000
```

OCTAL ADDITION

Octal addition is performed in much the same manner as decimal addition. A sum and carry technique is used, and the sum and carry are determined by reference to an addition table. The following OCTAL ADDITION TABLE may be used until you are familiar with octal addition:

		ADDEND							
		0	1	2	3	4	5	6	7
AUGEND	0	0	1	2	3	4	5	6	7
	1	1	2	3	4	5	6	7	10
	2	2	3	4	5	6	7	10	11
	3	3	4	5	6	7	10	11	12
	4	4	5	6	7	10	11	12	13
	5	5	6	7	10	11	12	13	14
	6	6	7	10	11	12	13	14	15
	7	7	10	11	12	13	14	15	16

Addition is merely a quick method of counting, and you are already acquainted with octal counting. You recall that 2 + 5 really means "counts five numbers beyond "2". As you know, symbols for quantities up to and including "7" are identical in the decimal and octal systems; consequently, additions resulting in a sum of "7" or less are the same in both systems. When the sum exceeds "7", however, the results of decimal and octal additions are different. Thus $5_{(8)} + 2_{(8)} = 7_{(8)}$, but $7_{(8)} + 1_{(8)} = 10_{(8)}$. If you increase the digit "7" by one count and if there are no single symbols for the quantities "8" and "9", you reach "$10_{(8)}$". In other words, the radix of the system has been exceeded; therefore, the original counting column goes to "0" and a carry of "1" is generated in the adjacent column. As you recall from decimal arithmetic, you can solve any problem if you have learned the sums of each possible pair of single-symbol decimal numbers. Thus you learned such sums as $6_{(10)} + 3_{(10)} = 9_{(10)}$; however, in octal additions you must learn such sums as $7_{(8)} + 7_{(8)} = 16_{(8)}$ and $5_{(8)} + 5_{(8)} = 12_{(8)}$. These may look both strange and difficult to a person who is familiar with only the decimal system, but luckily there is an easy way to find the sum of two single-symbol octal numbers. Here is the "gimmick": To add any two single-symbol octal numbers, add them as decimal numbers; then if the decimal sum exceeds "7", add "2" to get the sum in octal. For example:

$6_{(8)} + 5_{(8)} = ?$
$6_{(10)} + 5_{(10)} = 11$ (sum exceeds "7")
therefore 11 + 2 = 13

Study the following examples of octal addition:

7	7	7	7	7	7	7	7
+0	+1	+2	+3	+4	+5	+6	+7
7	10	11	12	13	14	15	16

6	6	6	6	6	6	6	6
+0	+1	+2	+3	+4	+5	+6	+7
6	7	10	11	12	13	14	15

14	65	74	3647	27564
+34	+35	25	2514	30723
50	122	121	6363	60507

OCTAL SUBTRACTION

Octal subtraction, as in decimal subtraction, may be performed directly by a subtract and borrow routine. When the minuend is smaller than the subtrahend, a "1" must be borrowed from the adjacent left hand column. Rapid calculations in the octal system are possible only when the operator is thoroughly familiar with the associated addition and subtraction tables. The following abbreviated octal subtraction table is included for your convenience.

Study the following examples of octal subtraction:

7	10	46	260	600	1064
2	2	37	124	275	575
5	6	7	134	303	267

As you already know, programmers often use the octonary system to express information because octal notation is shorter and less cumbersome than binary. Therefore, you should become familiar with the "sevens" (modulus minus one) complement for octal numbers.

BINARY NUMBER	COMPLEMENT	OCTAL NUMBER	COMPLEMENT
000	111	0	7
001	110	1	6
010	101	2	5
011	100	3	4
100	011	4	3
101	010	5	2
110	001	6	1
111	000	7	0

Note the correlation between the three column binary complement and the single digit "seven's" complement. For example, the binary number 110 111 101 001 011 is equivalent to 67513 in octal; in complement form, the two numbers are represented as $001\ 000\ 010\ 110\ 100_{(2)}$ and $10264_{(8)}$, respectively.

To perform subtraction by the "seven's" complement method, proceed in the following manner:

1. Convert the first octal digit to binary:

   7632 0000 0000 0000 =
   (1.11_2) 632 0000 0000 0000

2. Now, the first bit is the sign bit and the first octal digit is 3 (or 11_2). It is probably just as easy to leave the first octal

OCTAL SUBTRACTION TABLE		MINUEND						
		10	11	12	13	14	15	16
SUBTRAHEND	7	1	2	3	4	5	6	7
	6	2	3	4	5	6	7	10
	5	3	4	5	6	7	10	
	4	4	5	6	7	10		
	3	5	6	7	10			
	2	6	7	10				
	1	7	10					
	0	10						

digit in binary. Be sure you add in binary when you add.

3. Follow the rules for BAD and BSU in Figure 23.

You can check the results by using the "borrow" method of subtraction.

OCTAL MULTIPLICATION

Octal multiplication may be performed in a roundabout fashion by converting the octal numbers to be multiplied to decimal, performing the required operation, and then converting back to octal. If, however, you wish to perform octal multiplication, you must first learn the octal multiplication tables. This requirement is analogous to your having learned the decimal multiplication tables prior to performing decimal multiplication.

The following abbreviated octal multiplication table is included for your reference and convenience:

X	OCTAL MULTIPLICATION TABLE							
	1	2	3	4	5	6	7	10
1	1	2	3	4	5	6	7	10
2	2	4	6	10	12	14	16	20
3	3	6	11	14	17	22	25	30
4	4	10	14	20	24	30	34	40
5	5	12	17	24	31	36	43	50
6	6	14	22	30	36	44	52	60
7	7	16	25	34	43	52	61	70
10	10	20	30	40	50	60	70	100

The operations used in octal multiplication are similar to the operations used in decimal multiplication. The multiplicand is multiplied by one digit of the multiplier at a time to form a series of partial products that must be added to obtain the final product. The digit-by-digit multiplications are performed using products given in the octal multiplication table, and the sums are obtained using the octal addition table. The position of the octal point is determined exactly as it is in decimal multiplication.

Study the following examples of OCTAL multiplication:

$$\begin{array}{r} 462_{(8)} \\ \times\ 35_{(8)} \\ \hline 2772 \\ 1626 \\ \hline 21252_{(8)} \end{array} \qquad \begin{array}{r} 67.2_{(8)} \\ \times\ 1.04_{(8)} \\ \hline 3350 \\ 000 \\ 672 \\ \hline 72.550_{(8)} \end{array} \qquad \begin{array}{r} 354_{(8)} \\ \times\ 65_{(8)} \\ \hline 2234 \\ 2610 \\ \hline 30334_{(8)} \end{array}$$

OCTAL DIVISION

Octal division is performed in the same manner as decimal division except that the octal division and subtraction tables are used in place of decimal tables. However, you may find it less difficult to perform octal division in a more roundabout fashion--that is, first convert the octal division and dividend to decimal numbers, perform the required operation in decimal, and finally convert the decimal quotient back to octal. Study the following example of octal division:

$$\begin{array}{r} 462_{(8)} \\ 35_{(8)}\ \overline{)\ 21252_{(8)}} \\ 164 \\ 265 \\ 256 \\ 72 \\ 72 \end{array}$$

The MSD of the quotient is generated by examining the division of $212_{(8)}$ by $35_{(8)}$ and by deciding, on a trial basis, the largest number that $35_{(8)}$ can be multiplied by (resulting in a product less than 212). Thus, $4_{(8)} \times 35_{(8)} = 164_{(8)}$. The subtraction of the product ($164_{(8)}$) from $212_{(8)}$ is performed by direct octal subtraction. The process is continued until the required number of octal digits have been generated.

GENERAL

There are three basic methods for representing data in a computer--that is, coded, numeric, and logical. The question of how this information is arranged for presentation to the computer may be puzzling. In this discussion, we will find that data may be represented on various recording media by a Boolean code or by directly readable characters. It is known that information in the form of arranged patterns of "0's" and "1's" is coded for use in the computer. This unit of information arrangement and presentation is the computer word.

A computer word is of definite size; i.e., it consists of an exact number of binary symbols, each of which is termed a bit (in a binary machine). Each computer has its own word size which is of fixed length and arrangement. Some computers have been designed to handle computer words of 40 bits; others may use words of 30 bits or less. The number of bits in a computer word is expressed as its length. The length or size of the computer word makes available a definite number of positions for coding information in binary form. The choice of word size is not arbitrary. There is an optimum word size for any digital computer which is related to the "average" problem to be solved by the computer, the number of digits in the instruction code, and by the degree of desired precision in computation. In general, the design of computer equipment to reflect a specific length of the computer word is related to requirements.

There are two methods of recording alphanumeric characters which can be directly read by man. One method employs a photo sensing device to optically read characters directly from statements or bills. The second method uses a magnetic sensing device to read characters printed in magnetic ink.

Specially designed characters must be used by both the optical and the magnetic ink systems. The designs of these characters are similar to normal print characters but they are unique enough to allow rejection of normal print characters. Examples of these characters can be seen in Figure 24.

CODED METHOD

BINARY-CODED DECIMAL (BCD)

The reader has undoubtedly spotted one flaw in the binary system. Few human beings think in terms of binary numbers. Even the highly trained computer specialist pays his bills and counts his money in the decimal system. The binary number 1111111111 is equivalent to 1023, but the binary number 11111111111 is 2047 - significantly different, but difficult to recognize in binary form. How difficult it would be to interpret a binary number such as 1101001010, but its decimal equivalent, 842, is immediately recognizable to everyone.

There are many computer number codes that use modifications of the straight binary code. One of the most useful of these is the binary-coded decimal (BCD) code, which is a combination of the binary system and the decimal system. The BCD code has many useful advantages in data processing computers. All the arabic (decimal) numerals from 0 through 9 can be defined by four binary bits, as shown in Figure 25.

BINARY	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

FIGURE 25
Binary-Decimal Conversion

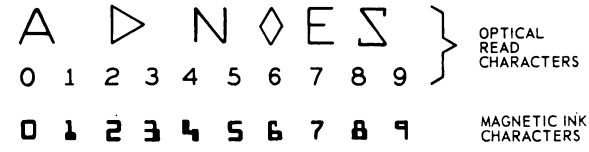


FIGURE 24
Special Characters

In Figure 25, we have used the conventional method of symbolizing conditions through the use of "1's" and "0's". In each binary number, the bit position on the extreme right tells whether the basic unitary increment (1) is included in the number. The second column from the right represents 2^1 (or just 2); the third column represents 2^2 or the value 4, and the fourth column represents 2^3, or the value 8. Thus, the decimal number 7 is made up of 1 + 2 + 4 =7; the decimal number 9 is made up of 1 + 8 =9, or $2^0 + 2^3 = 9$.

The binary-coded decimal system does waste some possibilities. The binary numbers for 10, 11, 12, 13, 14, and 15 can also be represented by four bits, but they are not normally used in the BCD system. To indicate numbers greater than 9, another group of four bits is used to represent each additional digit. For example, the number 34 is represented in BCD as:

0011	0100
3	4

Each decimal digit is thus represented by four binary bits in the BCD system. Although this system is not as efficient as straight binary, it has the advantage that it can be interpreted at a glance. Conversion to straight decimal notation for activating input-output equipment that accepts only decimal information is relatively simple. The BCD system is significantly more efficient than the decimal system, since each digit is specified by four bits as opposed to ten for the straight decimal system.

HOLLERITH CODING

The computer is not confined to directly representing real numbers in binary form. Binary words can also be coded to represent one or more alphanumeric characters. There are many types of alpha numeric codes used in the wide variety of data processing systems in use today. However, in this discussion, we will deal with only two of the most commonly used alphanumeric codes--12-Bit and 6-Bit Hollerith.

Figure 26 may be used to illustrate the manner in which data is recorded on punched cards in 12-Bit Hollerith Code.

There are 80 vertical columns on each card, and each column intersects each of the 12 horizontal rows. A hole punched at one of these intersections represents a "1" bit, and a "no hole" condition represents a "0" bit. A decimal number from 0-9 can be represented in any one column by one hole punched in the appropriate row of that column. An alpha character can be represented by a combination of two holes punched in one column. Finally, special characters can be represented by three holes in a column; one in the zone row, and two in the digit rows. The punched code for any alphanumeric or special character can be determined by use of the chart in Figure 27.

In Figure 26, the characters "ALE" in columns 1-3 denote the following:

Hollerith Code:	A =12, 1 punches
	L = 11, 3 punches
	E = 12, 5 punches
Binary Code:	A = 000010000001
	L = 000000100010
	E = 000000001001

In Figure 27, the Hollerith Code shown on the chart is the same for numbers and letters in any system; however, the special symbol codes might vary from system to system.

An obvious disadvantage of the 12-bit Hollerith code is its low packing efficiency. The term packing means to include several short pieces of information into one computer word. For example, a computer with a 48-bit word can hold only four 12-bit Hollerith characters. To overcome this packing problem, a 6-bit Hollerith code can be used to represent a maximum of 64 characters. The typical 6-bit code shown in Figure 28 is divided into alpha and BCD parts. The

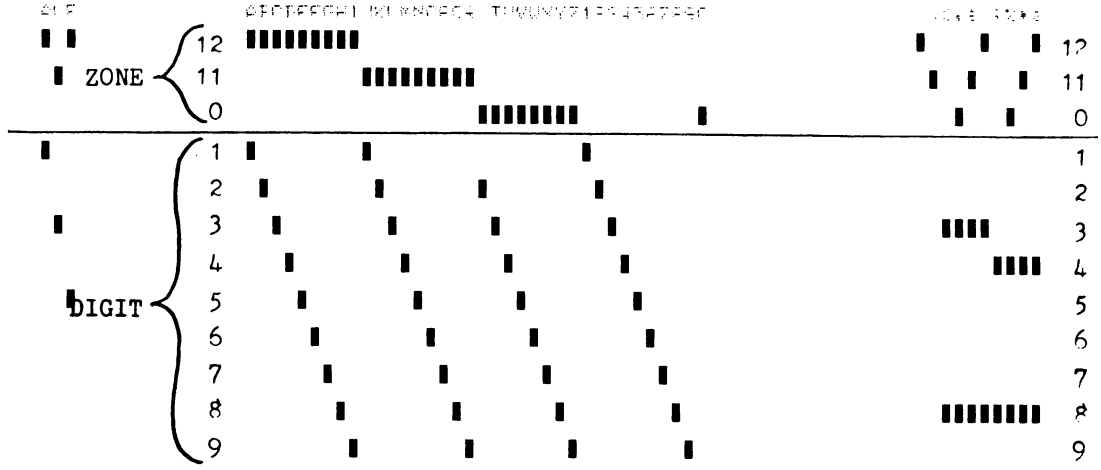


FIGURE 26
Punched Card Data Storage

12 Bit ZONE ONLY	NO ZONE	12	11	Ø
1	1	A	J	/
2	2	B	K	S
3	3	C	L	T
4	4	D	M	U
5	5	E	N	V
6	6	F	O	W
7	7	G	P	X
8	8	H	Q	Y
9	9	I	R	Z
8-3	+	.	$	,
8-4	-	□	*	%

FIGURE 27
Hollerith Type Wheel Codes

least significant digits comprise the BCD part and the two most significant digits make up the alpha part.

This code is called 6-bit Hollerith and is used by the BUIC System Computer, D-825, to code alphanumeric characters for storage on symbolic tapes. Other systems use variations of this code for magnetic tape, magnetic disk, and magnetic ink recordings.

Although the 6-Bit Hollerith code shown in Figure 28 represents 43 different characters, it should be noted that a maximum number of characters which could be represented is 2^6, or 64 characters.

BCD LSD \ ALPHA MSD	0 0	0 1	1 0	1 1
0 0 0 0	Blank	+	—	Ø
0 0 0 1	1	A	J	
0 0 1 0	2	B	K	S
0 0 1 1	3	C	L	T
0 1 0 0	4	D	M	U
0 1 0 1	5	E	N	V
0 1 1 0	6	F	O	W
0 1 1 1	7	G	P	X
1 0 0 0	8	H	Q	Y
1 0 0 1	9	I	R	Z
1 0 1 0				
1 0 1 1		.	$	,
1 1 0 0			*	

FIGURE 28
6 Bit Hollerith Code

There are sixteen possible combinations of the BCD bit ranging from 0000 to 1111, but only ten of these combinations are used to represent the decimal numbers from 0-9. The two bits which comprise the alpha part of the code can represent only four different conditions by themselves; however,

when combined with the BCD component, the number of characters which can be represented is increased by a factor of four.

Data is coded in 6-bit Hollerith primarily for symbolic storage on magentic tape and for printing messages out on computer-controlled typewriters, whereas data to be recorded symbolically on punched cards is coded in 12-bit Hollerith.

NUMERIC REPRESENTATION

BINARY

It is known from actual experience that the amount of equipment required for a digital computer depends on the base of the number system utilized by the computer. Most modern computers are designed to use the binary system (base 2) because this system is the most efficient with present-day components which are binary in nature.

Thus, it stands to reason that unless numerical expressions are scaled down in physical size and expressed to some power, all arithmetic operations must be governed by the computer word size. Normally, the first bit of a computer word designates whether the binary number is positive ("0") or negative ("1").

SCALING

Digital computers manipulate only pure numbers; therefore, when numbers representing physical quantities are used as operands in a digital computer, the programmer must keep a record of the units in which the quantities have been measured. For example, in computing a velocity, the programmer must be aware that the quantities representing distance and time are expressed in miles per hour, respectively, in order to conclude that the result of the computation expresses velocity in miles per hour.

In the process of digitial computation, three things must be known about each quantity used as follows:

1. The digits in numerical expression of the quantity.

2. The position of the radix, or digital point, with respect to the digits.

3. The units in which the quantity is expressed (as discussed above).

Automatic digital computers, in general, keep no record of the units associated with the numbers they process, and the record of this information is therefore the responsibility of the programmer. Some digital computers are built to keep an automatic record of the position of the point in every number taking part in the computation. In such a computer, the point will automatically assume a position that depends on the result of the computation. A computer that does this is called a "floating point' computer. For reasons that will be evident below, a computer that keeps no record of the position of the point with respect to the digits may be called a "fixed point" computer. When programming a fixed point computer, the programmer must not only remember the units in which the quantities are expressed, but must also keep a record of the position of the point. The technique by which he does this is called "scaling." The advantage of floating point computation is offset by the disadvantage of using digits within the computer words to indicate the position of the point, thereby reducing the precision that can be achieved with a given word size. Therefore, scaling a number is altering the units in which mathematical variables are expressed, in order to bring all quantities within a given range.

If the number $.00132_{(10)}$ is scaled to a whole number, the result would be 132×10^{-5}. The decimal point would be positioned to the right of the LSD. In the above example, that would require moving the radix point five places to the right.

Examples:

$$.00132_{(10)} = 132 \times 10^{-5}$$

$$13200_{(10)} = 132 \times 10^{2}$$

The rule for moving the radix point is as follows: LARS

When moving the radix point to the left, add - LEFT ADD

When moving the radix point to the right, subtract - RIGHT SUBTRACT

When scaling a number, the resulting figure must be specified in the directions; however, there are special methods of scaling which require the same number format for a result. Normalization is one method, and applying the principle of scientific notation is another.

NORMALIZATION. The normalization method is merely scaling with the radix point to the left of the MSD.

Examples:

$$.00132_{(10)} = .132 \times 10^{-2}$$

$$13200_{(10)} = .132 \times 10^{5}$$

SCIENTIFIC NOTATION. Scientific notation is the other scaling method to be discussed. The radix point is usually placed to the right of the MSD when scaling, using the principles of scientific notation.

Examples:

$$.00132_{(10)} = 1.32 \times 10^{-3}$$

$$13200_{(10)} = 1.32 \times 10^{4}$$

The process of computer normalizing or normalizing to a floating point computer commands the conversion of the number to be normalized to a binary number (if in another system other than binary) before the normalizing process is accomplished.

Example:

$$.00132_{(8)} = .000000001011010_{(2)} =$$

$$.1011010 \times 2^{-8} \text{ or } .55_{(8)} \times 2^{-8}$$

If there had been a base (not to be confused with the term for radix) for a floating point machine, the suffix (in the above example) x 2^{-8} would be added to that base number. The exponent will always be a decimal number since the number of places the point is moved is counted in decimal; therefore, when added to a base number of any numbering system other than decimal, the exponent has to be converted to that number system.

Example:

Given a base of $200_{(8)}$, computer normalize the number $.00132_{(8)}$: $.00132_{(8)} = .000000001011010_{(2)} = .101101 \times 2^{-8}$; $-8_{(10)} = -10_{(8)}$ added to the base $(200_{(8)} + (-10_{(8)})) = 170.55_{(8)}$.

D-825 FLOATING POINT FORMAT

The D-825 uses a base of zero and reserves the most significant 12 bits of the computer word for its 12 bit SIGNED exponent. The least significant 36 bits represent a 36 bit signed mantissa. The mantissa indicates a binary quantity, and the exponent denotes the number of times the mantissa is raised by the power of 2. As a result, the exponent indicates the actual position of the binary point in the mantissa.

Example:

positive exponent, negative mantissa:
$0005\ 6234\ 0000\ 0000 = -22.34_8$

negative exponent, positive mantissa:
$4005\ 2234\ 0000\ 0000 = +.01116_8$

Use of the four basic floating point instructions in the D-825 facilitates operation with much larger numbers, and reduces considerably the amount of scaling which must be done by the programmer.

The building blocks of a digital computer are its individual circuits--hundreds, often thousands of them--interconnected to accomplish the operations of transferring and processing data. There are not this many different circuits, but a few basic types used again and again in various combinations. Before discussing these basic computer circuits and other devices, certain fundamentals must be examined, such as the types of electrical signals commonly used and the nature of the simple logic operations performed by the circuits.

INFORMATION SIGNALS

The transfer and processing of information in a digital computer is done by switching and storing information signals--electrical signals representing numbers. Most of the common electronic parts (relays, vacuum tubes, magnetic cores, etc.) perform excellently in bistable (two-state or on-off) operation. Because of this, it is usually easiest to make computers work internally in the binary number system. In this case, the information signals must represent the binary digits, "1" and "0".

There are several possible ways of representing the binary "1's" and "0's" electrically. For example, when it is necessary to transfer numbers over a single signal line between circuits, as shown at (a) of Figure 29, the easiest method is to place a d-c voltage on the line to represent a binary "1" but no voltage to represent a "0".

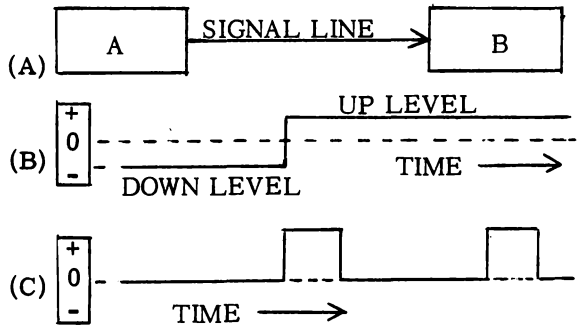


FIGURE 29
Common Number Signals.

VOLTAGE LEVEL REPRESENTATION

An alternative to this voltage-or-no-voltage method uses a steady-state, positive d-c voltage for a "1" and a negative (or less positive) for a "0". This method of representing numbers with d-c levels is shown at (b) of Figure 29. Circuit "A" indicates a "0" to circuit "B" by holding the signal line at the down level. When it is desired to signal a "1", the level is brought up to a positive potential and held there as long as necessary. As long as the level is up, a "1" is present at the input to circuit "B"; when it is down, the input is "0".

PULSE REPRESENTATION

One characteristic of the voltage level information signal is that it can be held up or down as long as necessary. In many cases, however, all that is needed is a "1" signal of very brief duration to trigger the following circuit, so a pulse can be used to represent a "1", as in (c) of Figure 29. If a pulse (positive or negative) represents a "1", then the absence of a pulse logically represents a "0".

Using the method of pulse signals, the output line from circuit "A' remains at some reference level until a "1" must be transmitted to circuit "B", whereupon a signal pulse is generated by "A" and is placed on the line. The pulse appears at the input of circuit "B", signaling a "1", and quickly dies out. A typical pulse used to represent a "1" might be only 0.1 microsecond (μsec) in duration.

TRANSMISSION METHODS

The voltage levels and pulse signals are the two basic types (but not the only possible ones) used to represent numbers (coded information) in digital computers. But a single level or pulse represents only one binary bit, yet the computer must work with long binary numbers (many bits), or words. Let us look at how these computer words are transmitted from one part of the machine to another.

PARALLEL. If one signal line between two circuits can, at a given moment, transmit a "1" or a "0", it is reasonable to conclude that a complete, five bit word requires five lines in parallel. This is called parallel transmission. The binary number 10011 is shown in Figure 30 as it would be sent in parallel form, with pulse-type signals.

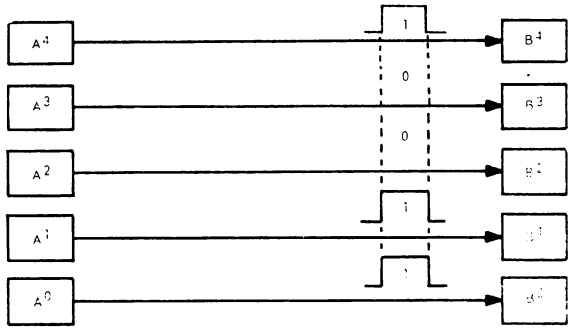


FIGURE 30
Parallel Transmission of Numbers

SERIAL. In addition to the parallel (or side-by-side) method just described, there is one other basic way of transmitting number signals. This second method also uses either levels or pulses, but sends the bits of the number, one after another, down a single line. Thus, the bits are sent in sequence, or serially, so this is called serial transmission. In the serial method, it is usual to send the least significant bit of the number first, followed by the other bits in order of increasing significance. This makes sense when it is remembered that addition, subtraction, etc., are performed bit by bit in the same order. The binary number previously used as an example, 10011, is shown in Figure 31 as it would be sent in serial form, with pulse-type signals. The number is sent from circuit "A" to circuit "B" as a train of pulses and no-pulses. The first bit transmitted by circuit "A" and received by circuit "B" is the least significant bit, followed by the remaining bits in order of their significance. There must be spacing between successive pulses, otherwise they could not be distinguished by the receiving circuit.

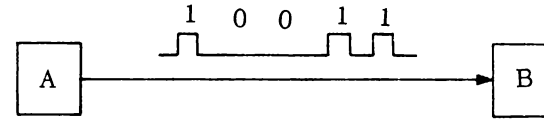


FIGURE 31
Serial Transmission of Numbers

COMPARISON OF METHODS. The principal differences between the parallel and serial methods of transmission show up in a comparison of Figures 30 and 31. In the parallel method of Figure 30, five sending and five receiving circuits are involved to handle a five-bit number. In the serial method (Figure 31), only two circuits are needed--one sending, the other receiving. On the other hand, all bits are sent simultaneously in the parallel method, so the entire number is transmitted in the time it takes to send only one bit. In the serial method, the entire number is not known until all five bits have been sent, one after the other, so it takes five times as long to send the same complete number.

TIMING

Another matter of importance in transmitting numbers is timing. We will now take a look at the timing requirements for the two transmission methods just explained.

PARALLEL TRANSMISSION. Consideration of the parallel method shown in Figure 30 indicates that simultaneous operation of the circuits signaling a given number is a must if circuits "B^0" through "B^4" are going to operate on the number as soon as it is received. If the number signals were voltage levels, for example, and the signaling circuits "A^4" and "A^0" were operated together, followed a moment later by signaling circuit "A^1", the receiving circuits would first get the number 10001, which would then change to 10011. With pulse-type signals of short duration, the same operation of the signaling circuits would send 10001 and then 00010. Either of these occurrences, resulting in incorrect numbers getting in, could cause errors in an arithmetic machine. Therefore, it is important in parallel transmission to time all of the bits of a number to arrive simultaneously at their destinations.

SERIAL TRANSMISSION. Timing is also vital in serial transmission, as an examination of Figure 31 will indicate. Since the bit signals are sent down a single line, one after another, some rigid timing system is a necessity, especially with pulse-type signals. If the signals were sent at varying intervals, the receiving circuit would have no way of telling whether a long space between pulses was a "0" or merely spacing. Furthermore, if voltage level signals were used, the receiving circuit could not distinguish between a "1' and two consecutive "1's" or between a "0" and two consecutive "0's". Consequently, the timing of serial transmission must also be controlled. This is done by establishing the period of time necessary to send one bit, which is the smallest piece of information handled in the computer.

Once the time period for transmission of a single bit (called one bit-time) has been determined, the problem of serial timing is handled by rigidly controlling the length and spacing of the bit signals in every number (computer word) transmission. Figure 32 shows an example of the timing of both the pulse-type and level-type signals making up the number 10011. In the pulse system, the relative durations of the pulses and spaces vary from one computer to the next. In some systems the space is the same width (in time duration) as the pulse.

In many computers, the basic source of timing or synchronizing signals is the CLOCK, usually a pulse generator which, controlled by an accurate oscillator, puts out a continuous string of rigidly timed pulses. The clock pulses are generated one per bit-time and are sent to all parts of the computer to control the transmission of numbers and the timing of operations. Thus, in a computer using a one microsecond bit-time the clock must generate one million pulses per second, at exact one microsecond intervals, from the time the computer is turned on until it is shut down.

SWITCHING LOGIC

Up to this point, the basic types of signals used to represent information in digital computers and the basic methods of moving the information from one part of a computer to another have been covered. It has been mentioned that all the arithmetic and other operations performed in a digital computer are done by switching and storing information (in the form of numbers) in the proper combinations and sequences.

The operations carried out by a digital computer are operations of logic. Arithmetic--in fact all mathematics--is rigidly based on logic; in other words, arithmetic is a systematic process of manipulating numbers involving simple operations carried out according to precise rules. If numbers are to be represented by voltage levels and pulses, some system of manipulating these voltages according to the logical rules of arithmetic must be found. Circuits which accomplish this function in a computer are called logic circuits.

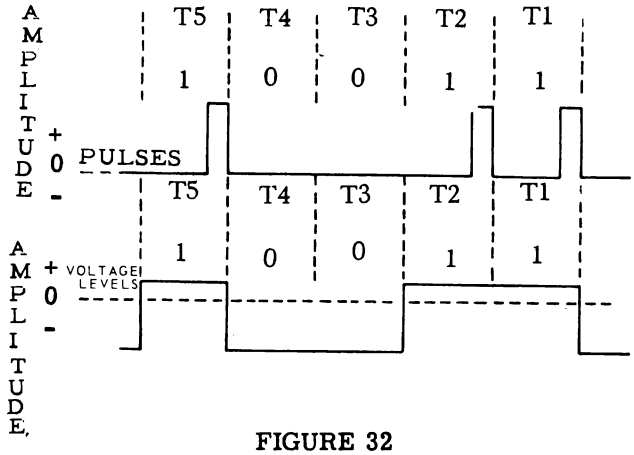


FIGURE 32
Timing of Serial Numbers

LOGIC OPERATIONS

Let's now take a look at how switching is applied in operations of logic. This can best be understood by looking at the types of logic operations that can be performed easily by a switching circuit.

OR LOGIC. One of the common logic operations is the alternative or choice, called the OR function. This comes into play whenever any one of two or more alternate possibilities can bring about a specified result. For example, "We'll go to the movies if George, Peter, or Joe shows up." In this case, the arrival of George OR Pete OR Joe leads to the result, movies. This can be written in shorthand form:

George OR Peter OR Joe =Movies

This situation can also be symbolized in diagram form, as shown in Figure 33a. The label in the block indicates that OR is the relationship between its "inputs", which are, of course, the arrival of George, Peter, or Joe. Another way of thinking of it--more accurate when dealing with equipment--is that the block applies the OR function to its inputs. The block produces an "output"--movies--only when the inputs meet the OR requirements: in other words, when at least one of the inputs appears. This diagram can be altered, as in Figure 33b, to illustrate the general case, any OR situation. Three inputs are shown, although any number except one is possible (one condition offers no alternative, hence no OR). The OR function produces a specified result, D, when any one of its input conditions, A OR B OR C, is satisfied. Notice that if any two, or even all three, of the inputs appear together, the output is still produced.

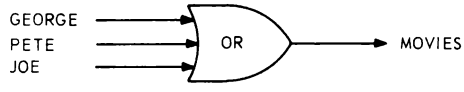


FIGURE 33a
OR Situation Symbolized

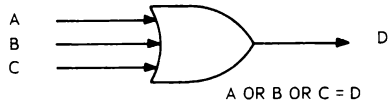


FIGURE 33b
OR Function

AND LOGIC. The AND function is another common logic operation. It requires that all of its two or more possible conditions (inputs) be present at the same time to bring about a specified result (output). For instance, "You need inductance and capacitance and resistance to build a bandpass filter." All three are required--and all at the same time--to produce the result, a filter. If any one is missing, or if the three are present only at different times, the specified result is not produced. The logic can be written:

L AND C AND R = Filter

Figure 34 illustrates the AND function in diagram form. Again, any number of inputs except one is possible. The AND function produces a specified result, D, when all its input conditions, A AND B AND C, are fulfilled at the same time.

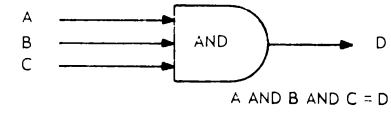


FIGURE 34
AND Function

NOT LOGIC. Another logic operation of importance is the NOT function, called inversion. Inversion means a turning upside down or a reversing of relationships. In working with two-valued logic, this means changing every quantity to its opposite. Every "yes", when inverted, becomes a "NOT yes", which is the same as a "no". Similarly, a "no", inverted, becomes a "yes". Figure 35 shows the symbols for the NOT function.

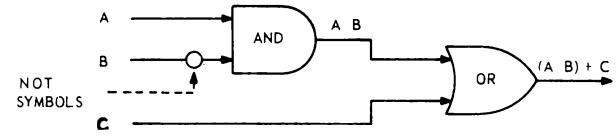


FIGURE 35
NOT Function

For example, someone might say, "I'll go if Tom does, but not if it rains." Examination shows that this involves an AND function and a NOT function.

Tom goes AND (NOT Rain) = I Go

This can be diagrammed with an AND block and an inverter, as shown in Figure 36; the combined functions are often called the AND NOT. (An OR NOT arrangement can be put together in similar fashion from an OR and an inverter.) Notice that if the inverter input (rain, in this case) is present, it prevents the AND from producing an output. The presence of an inverter input means no inverter output; hence, a missing input to the AND. The AND cannot operate unless all its inputs are present simultaneously. This prevention of the AND operation is called inhibiting which, used this way, means about the same thing as prohibiting.

The OR NOT function by itself is drawn as shown in Figure 37. Operating in OR fashion, the inhibitor produces an output, C, when inputs A OR B are present.

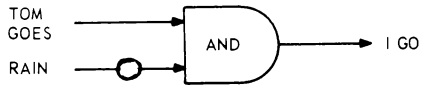


FIGURE 36
AND NOT Diagrammed

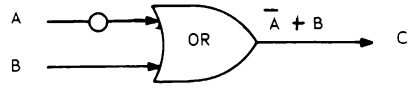


FIGURE 37
OR NOT Function

CIRCUIT LOGIC

Now that the basic logic functions used in digital computer circuitry have been examined in terms of information only, it is time to see how physical circuits operate according to the rules of these functions. The inputs are now going to be electrical signals representing the facts or events that must be logically connected. The logic blocks previously used to diagram the functions are henceforth actual physical circuits. And, finally, each output is an electrical signal representing the specific result of applying the rules of a particular logic function to a particular set of inputs. To put it another way, each output is a logical conclusion.

To see how a switching circuit performs a logic function, consider the case of a home owner who wants to be warned when someone comes to either his front or back door. This involves the OR function, and the logic of this situation can be diagrammed as shown in Figure 38, using the simple OR block. The ordinary manner of "solving" this, of course, is to install a doorbell circuit, with a pushbutton switch at the front door and another at the back and a bell inside the house. A battery can be used to power the bell, as shown in Figure 39.

The pushbuttons are not parts of the logic circuit, but are simply devices to translate physical facts or events into electrical signals. They put the information into the circuit. The fact, "somebody at the front door", is translated to a voltage of six volts when that "somebody" presses the front-door button. The voltage, which can also be considered as a binary "1", is applied to one input of the OR circuit. According to the OR function rule, an output is produced when one input OR the other is present. So a binary "1" at the front-door input results in an OR circuit output that rings the doorbell. Following this reasoning, a binary "1' at either

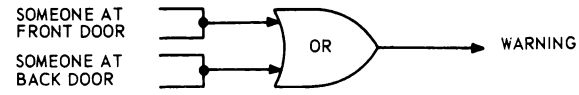


FIGURE 38
Logic of Doorbell Situation

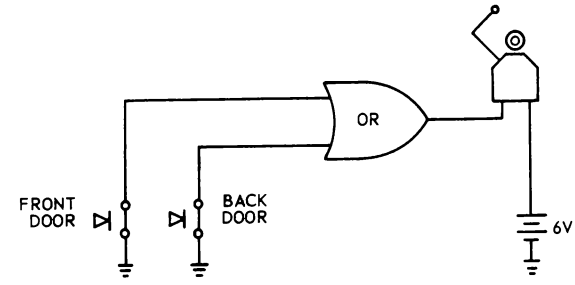


FIGURE 39
Doorbell Circuit, Showing Logic

OR circuit input represents "somebody present", so a binary "0" (no voltage, or zero volts) must represent "somebody NOT present".

"1" = Somebody present
"0" = Somebody not present

Thus, electrical signals can be made to represent the binary numbers which in turn are made to represent specific items of information. The six volts can be thought of as the up level voltage, in which case zero volts is the down level voltage.

Now, the OR circuit itself, inside the block in Figure 39, must be constructed to operate in accordance with the rules (logic) of the OR function; in other words, it must be built to produce an output when a binary "1" (up level voltage) appears at one input OR the other. What must the output be? Well, the bell must ring when somebody is present (at either door); binary "1" represents "somebody present", so the output must be a binary "1" or an up level of six volts. The current that flows as a result of applying this up level is capable of ringing the bell, so the choice of output is logically and electrically satisfactory. The bell can be considered as a device to transfer the information "somebody present" to the homeowner.

When there is nobody present at either the front or back door, a binary "0" (down level) is present at each OR circuit input. In this case, the output must also be a binary "0", or down level, representing "somebody not present". The down level cannot cause the bell to ring. The conditions of this situation are so simple it is apparent that the OR circuit itself need be nothing more than a parallel connection of wires from the pushbutton switches, as shown in Figure 40. Notice that, however simple it may be, this is the only part of the circuit that fulfills, by itself, the requirements of the OR function. It is the parallel method of connection that offers alternate input possibilities, making this an OR circuit. It is important to understand this distinction, although in practice it is common to speak of the entire parallel circuit, including the switches as the OR circuit.

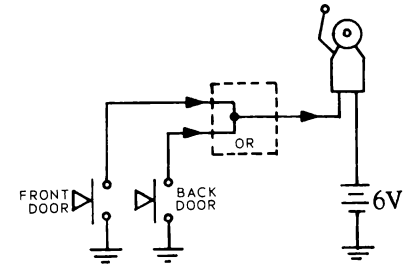


FIGURE 40
Complete Doorbell OR Circuit

This doorbell OR circuit provides a simple illustration of the manner in which a logic operation is carried out by an electrical or electronic circuit. Regardless of the type of logic circuit, two-valued information is represented by binary numbers "0" and "1", which in turn are represented by electrical signals. Means or devices are provided to get these signals into the circuit at the proper place and time. By building the circuit to operate upon the electrical signals according to the rules of the desired logic operation, the resulting output signals represent logical decisions or conclusions reached in accordance with the built-in rules. To be useful, these outputs are transmitted or transferred either to some other circuit or out of the computer.

So the computer logic circuits cannot "think' and do not know what information their inputs or outputs represent. There is nothing miraculous about them. They simply accept electrical input signals and operate with them in accordance with the circuit design, just as ordinary radio or TV circuits must do. All the thinking is done by the designers who build the rules of logic into the circuits and the programmers who direct the operation of the computer. The advantage of the complete computer is that it can perform a long sequence of these simple logic operations, at extremely high speed, by sending signals through a chain of logic circuits. By performing the proper sequence of operations, the computer does arithmetic.

COMPUTER LOGIC CIRCUITS

There are many different types of switching devices used in computer logic

circuits. Some of these devices are relays, crystal diodes, vacuum tubes, transistors, and magnetic cores. Each type device has certain characteristics, capabilities, and limitations which will be covered briefly at this time.

RELAY LOGIC CIRCUITS

Computers composed principally of relays are not often built today, due to the comparatively slow operate and release times of relays. However, some relay circuitry is often used in electronic computers, especially in the input and output elements.

CRYSTAL DIODE CIRCUITS

Crystal diodes are used mainly in logic switching operations. They offer the following advantages: very small, light in weight, require no heater power, very reliable, and have a longer life than vacuum tube diodes.

VACUUM TUBE CIRCUITS

Vacuum tubes are excellent switching devices, offering the advantage of high speed and the possibility of amplifying signals as they are switched. However, they have comparatively large space, power, and cooling requirements; these disadvantages indicate that the vacuum tube will see less and less use as newer devices are perfected.

TRANSISTORS

Although transistors are frequently thought of as replacements for vacuum tubes and can often be used in similar logic circuits, some newer types are well-suited to straightforward use as switches. Transistors offer several advantages for digital computer use. They are small and well-suited to miniaturized circuits, require little power, and dissipate little heat. As switches, they are as fast as vacuum tubes; hence, they can be used in high-speed computers to provide excellent reliability.

DELAY CIRCUITS

It is often necessary to delay a pulse signal in time as it is routed through a computer without affecting the width of the pulse or the time between pulses. This is the basic function of a delay circuit. The symbol for a delay line showing the effect it has on a signal is shown in Figure 41.

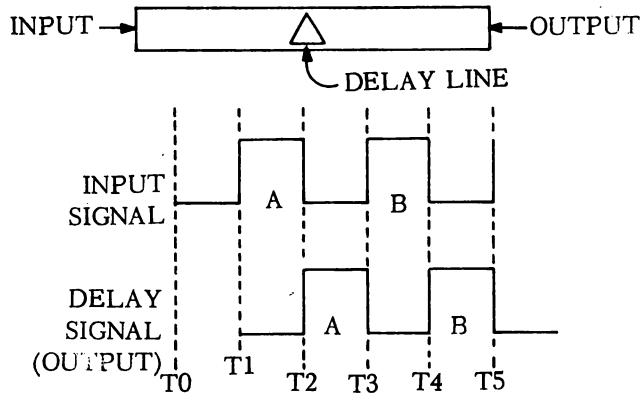


FIGURE 41
Delay Line Operation

MAGNETIC CORE LOGIC CIRCUITS

Magnetic cores, originally developed and widely used as storage devices, are now finding more and more use in switching and logic devices. The core, shaped like a tiny doughnut, is a bistable one-bit storage cell or memory. It is made of a material with magnetic properties.

At least three small coils are ordinarily wound on each magnetic core. Two of these are obviously for input and output of information; the third is needed for sensing or read out, of the information ("1" or "0") stored in the core. The basic, three-winding core is shown schematically in Figure 42.

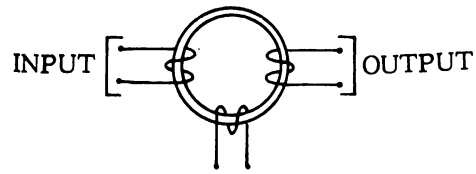


FIGURE 42
Magnetic Core

Since the core is made of a magnetic material, it is really a small magnet with the direction of flux (magnetic lines of force)

running either clockwise or counterclockwise inside the magnetic core. This direction of magnetization can be quickly reversed by applying a pulse to a coil wound on the core. Thus, by deciding that one direction of flux represents a "1", the other a "0", a pulse on the input coil can be made to insert a "1" which the core then stores because its direction of magnetization can be changed only from the outside. Inserting information into the core is called writing a "1" or "0" into the core, or setting the core to "1" or "0". Taking out the binary bit is called reading out, or simply readout.

To read information out of a core, a pulse is applied to the readout winding. The readout pulse always sets the core to "0". This is known as destructive readout. The readout winding may be called by various other names, such as reset, shift, or clear winding. If the core is already at "0" when the readout pulse is applied, no output pulse is obtained at the output coil. However, if the core is in the "1" state when the readout pulse is applied, the direction of magnetization of the core is changed to the opposite direction ("0" state) and a "1" pulse is obtained at the output coil. This output pulse can be used to set other cores or as an input to other logic circuits.

Another important characteristic of magnetic cores is that they will remain magnetized in the correct state even though the primary power to the computer is off for long periods of time.

SMALL-SCALE STORAGE CIRCUITS

The fact that all arithmetic and data-processing operations in a digital computer are accomplished by switching and storing electrical signals has been mentioned several times. The switching circuits that perform the logic operations have been examined briefly. It is easy to see that if signals representing "1's" and "0's" must be combined in certain logic circuits and if a signal available now is needed a little later, some means of "storing" this signal until it can be used is required.

A delay line type storage device works nicely for pulse signals and for brief storage periods of a few bit-times. However, this type storage device is not suitable when voltage levels are used or when the storage period is of either long or varying durations. It is impractical to send signals from all parts of the computer to the storage element each time bits of information are to be temporarily stored for use during later operations. What is needed is a small-scale, on-the-spot, bistable, storage device that can be set to "1" or "0" by the signal it receives. It must also be able to remain in that state, after the input signal disappears, until it is reset. It must, of course, be able to indicate its "1" or "0" state to other devices by means of one or more outputs. This indication may be continuous or it may be supplied only when demanded, as in the case of a magnetic core which indicates the bit stored only when a readout pulse is applied. With such a device it is easy to store a single bit until it is needed. When a parallel transmission is used, it is easy to store a complete word simply by providing storage places in parallel for each bit of the word. A group of devices for storing a complete word is called a "register".

BISTABLE CIRCUITS

With the possible exception of crystal diodes, all the switching devices used in logic circuits can be easily adapted to circuits for bit storage. Magnetic cores were originally developed for this purpose; their use in logic circuits came later.

FLIP-FLOPS. The flip-flop (also referred to as a "bistable multivibrator", "Eccles-Jordan circuit", "Trigger" or "toggle") circuit is a bistable device used to store a single bit of information. A fundamental characteristic of the flip-flop (abbreviated FF) is that at any given time it can be in only one of its two possible states.

Because the flip-flop circuit is stable in only one of its two possible states at a given time, and because it will remain in that state unless an input signal is applied, the flip-flop circuit can be used to store a

binary bit. One of the two states is designated the "ZERO" state and the other the "ONE" state.

Flip-flops have their own circuit symbols, which may resemble any of the three shown in Figure 43.

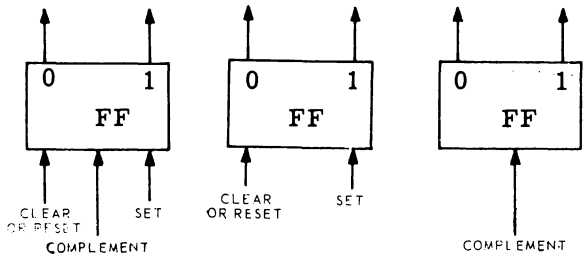


FIGURE 43
Flip-Flop Circuit Symbol

Another important characteristic of the flip-flop circuit is that the two outputs are always at opposite voltage levels. When one is high, the other is low. When one goes down, the other goes up. The inputs are normally pulses, although it is possible to use voltage levels. If a pulse is applied to the set input, the one side output goes high and the zero side output goes low. If a pulse is applied to the clear input, the zero side output goes high and the one side output goes low. If a pulse is applied to an input (clear or set) whose corresponding output is already high, the flip-flop doesn't change states. These rules of operation are illustrated in Figure 44.

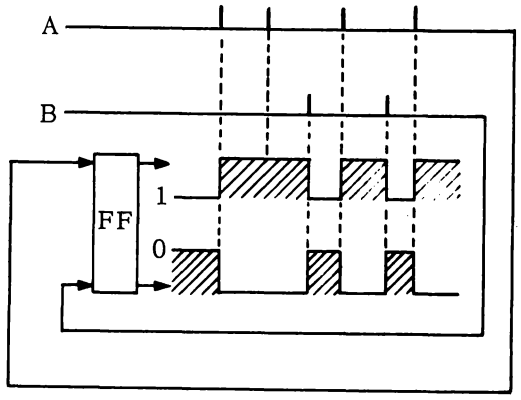


FIGURE 44
Basic Flip-Flop Operation

For some purposes, flip-flops require only the complement input. When a pulse is applied to the complement input, the flip-flop always changes its state.

BINARY COUNTERS. The term "binary counter" is nearly self-explanatory. It refers to a device, composed of appropriately connected flip-flops, which performs the function of counting in the binary number system. The counters count the number of input pulses and the stages of the flip-flops indicate the count binarily.

Counters are classified according to the function they perform; they are known as up-counters or down-counters. Up-counters count the number of input pulses. Down-counters subtract the input count, "count down", from some preset number.

Counters are also classified according to their circuit design, which may be serial or parallel. A serial counter consists of a group of flip-flops that binarily count a series of pulses. They are so connected that each flip-flop changes from the ZERO to the ONE state, or conversely, as it receives the output voltages pulse of the preceding flip-flop. The counter is used to serially count all pulses present at the input. Each pulse applied to the counter input changes the state of one or more of the flip-flops in such a way that the binary configurations in the flip-flops represent the number of input pulses counted. A four-stage serial up-counter which is capable of counting in binary from 0(0000) through 15(1111) is shown in Figure 45.

A parallel counter consists of a group of flip-flops connected so that the input pulse is simultaneously applied in parallel to each flip-flop. Parallel counters are used because their response time is much faster than serial counters, but they have the disadvantages of requiring more circuitry and consuming more power than serial counters.

STORAGE REGISTERS. A register, as mentioned earlier, is a group of storage devices used for storing a complete word. Four flip-flops connected as a 4-bit parallel storage register are shown in Figure 46.

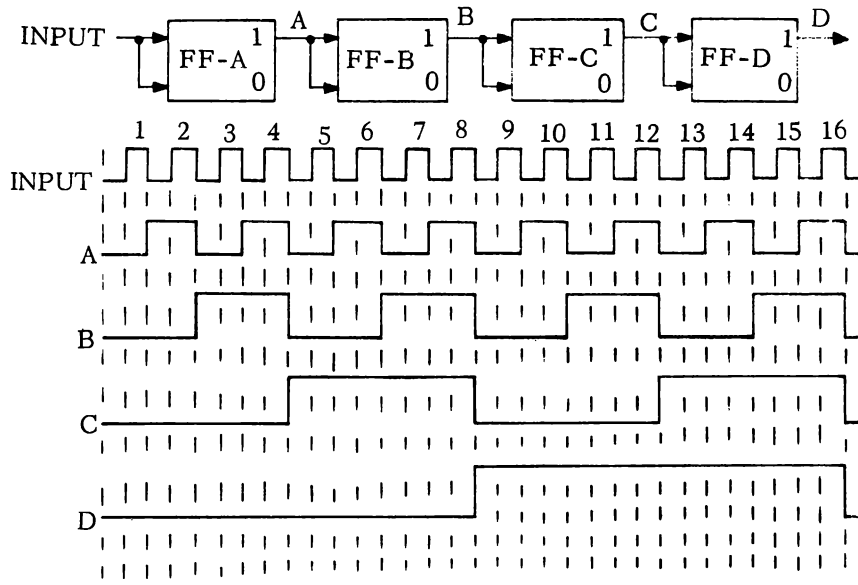


FIGURE 45
Four State Serial Up-Counter

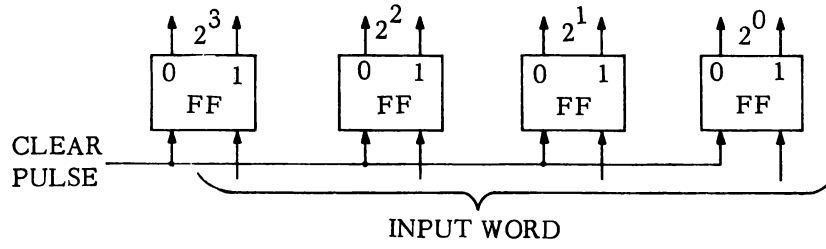


FIGURE 46
Parallel Flip-Flop Storage Register

The clear inputs are all connected in parallel so that a single pulse can be used to clear the whole register before the new word to be stored is applied. This clears all the flip-flops in the register to the "zero" state, wiping out any word that may have been stored previously. Then the word to be stored is applied in parallel form to the set inputs. The pulse in each bit position where there is a "1" sets the corresponding flip-flop to the "one" state. No pulses appear in the bit positions where there are "0", so the flip-flops in these positions remain in the "zero" state, and the correct word is stored.

SHIFT REGISTER. A shift register is built with the intention of shifting any numbers stored in it for a purpose other than that of ordinary storage. The purpose may be to convert words from serial to parallel form, from parallel to serial, or it may be to multiply or divide the numbers by some power of two.

The circuitry of a shift register is similar to that of a storage register in that a flip-flop is used to handle each binary bit of information. However, one additional input (called shift input) must be connected through additional logic circuits to each flip-flop. There are many possible arrangements of shift registers and which one will be used in a given case depends upon the exact operations to be handled by that shift register.

ARITHMETIC AND CONTROL

INTRODUCTION

The computer element which performs mathematical computations and data

manipulations is called the arithmetic element. The need for this element is self-evident; a means of performing arithmetic operations (addition, subtraction, etc.) is essential to the solution of any mathematical problem. In most large-scale computers, the memory element is completely separate from the arithmetic element; however, the two units are both interrelated and interconnected. By transmitting electric pulses back and forth, these two elements are able to communicate. The memory element sends information to the arithmetic element for processing, and the arithmetic element sends back processed information for storage. The arithmetic element handles and manipulates the numbers; however, the control element tells the arithmetic unit HOW and WHEN to handle them. The instructions to the computer are not in a form that will operate the arithmetic unit directly. To interpret the instructions for the arithmetic element is a major function of the computer's control element. Comparably, the operator of a desk calculator who first keys in the data and then presses the instruction key is exercising the same control function. However, in high-speed digital computers, the control task is performed electronically. Given the correct instructions and accurate data, there are few mathematical problems that a computer cannot solve; moreover, it is estimated that the AN/FSQ-7, a computer used in the SAGE Air Defense, averages less than one error in every ten billion operations.

COUNTING

A digital computer is built by assembling networks of basic circuits to perform arithmetic operations, to handle input and output information, and to control the internal working of the machine.

The ring-like nature of the counting process makes it simple to design networks that can count input signals. Such circuits, called "counters", are used for various purposes, such as counting steps in the program as they are executed. In any case, a signal--usually a pulse--is generated each time the event to be counted occurs. The counter then counts these pulses. For instance, a signal is sent to the "program counter" each time a step of the program is completed; the counter keeps track of the progress of the program.

ADDITION

Simple arithmetic addition is nothing more than a short-cut method of counting from smaller numbers to larger ones. Since the only binary digits are "0" and "1", binary addition is merely a matter of counting in columns and correctly handling the carries between columns. Although it is common practice in pencil-and-paper arithmetic to add a whole column of figures at one time, this practice has not been found practical in digital computers. Instead, a computer adds the first two numbers; then the third number is added separately to the sum of the first two; accordingly, the fourth number is then added to the sum of the first three numbers. This repetitive process is used to add columns of numbers of any length.

Consider the addition of two bits. We will identify the bits as bit A and bit B. Each of these two bits may be either 0 or 1. Under these conditions there are only four possible combinations of these two bits. They are 0 + 0, 0 + 1, 1 + 0, and 1 + 1. The table in Figure 47 shows each combination and the resulting sum and carry. This table is called a truth table from Boolean Algebra.

A computer circuit must now be found that produces the results shown in the truth table. There must be two inputs to this circuit, bits A and B. There must be two outputs, the sum and carry. The circuit in the computer that does this is shown in Figure 48. It is called a half-adder. The reason for this name will be obvious later. Each part of Figure 48 shows one of the four possible combinations of A and B and the resulting outputs. The various voltage levels throughout the circuit in each case are also shown. A "0" indicates a low voltage. A "1" indicates a high voltage.

Figure 48 shows that the half-adder is made of the same basic logic circuits as discussed in computer logic. The

	A	B	Carry	Sum	
(1)	0	0	0	0	0 +0 00
(2)	0	1	0	1	0 +1 01
(3)	1	0	0	1	1 +0 01
(4)	1	1	1	0	1 +1 10

FIGURE 47
Truth Table for the Addition of Two Bits

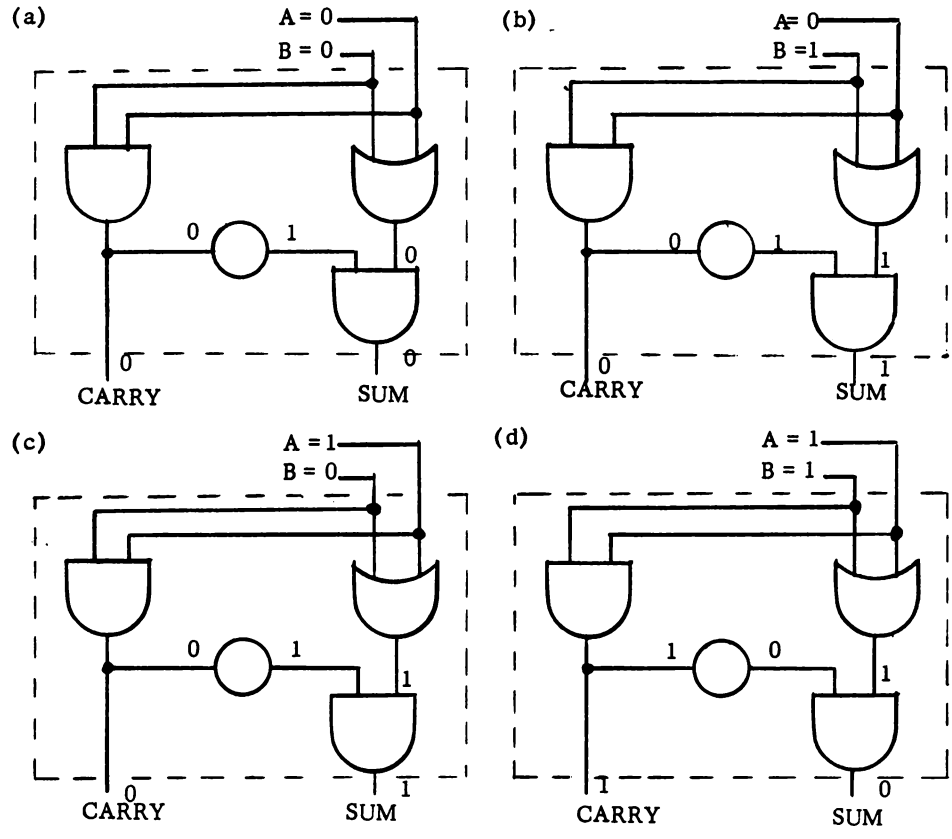


FIGURE 48
The Half-adder

half-adder is capable of adding any two bits and producing the correct sum and carry. However, if a computer could add only two bits, it would be extremely limited in its capacity.

The next step that we will take will be to double the computer's capability. A circuit which will add two binary numbers with two bits per number and produce the correct sum and carry is shown in Figure 49. This circuit is called a full-adder. Figure 49 shows the addition of two binary numbers, 01 and 11, as an example. Notice that the circuits has four inputs and three outputs.

The full-adder is required when it is necessary to add the two bits of a column and a carry from a previous column. In Figure 49 the half-adder is used to add the two bits in the first column. This is done because there will never be a carry in this column from a previous column.

To increase the number of bits per number that the computer can add, it is only necessary to add additional full-adders with an "OR" circuit between each full-adder as shown in Figure 50. One full-adder will be required for each additional bit in the numbers. Twenty-nine full-adders and one half-adder would be required to add two numbers with 30 bits per number.

We have shown the development of a computer that can add any two binary numbers. Basically, this is all that any computer can do. A computer multiplies, subtracts, and divides through adaptations of addition. If a computer subtracts, we have seen how it complements and adds. If a computer is to multiply two numbers such as 4 x 3, it could simply add 4 + 4 + 4 and get 12. Similarly, a computer could divide by repeated subtractions. So the computer is actually quite a simple device basically. It is only its tremendous speed and vast memory that make it seem so incredible.

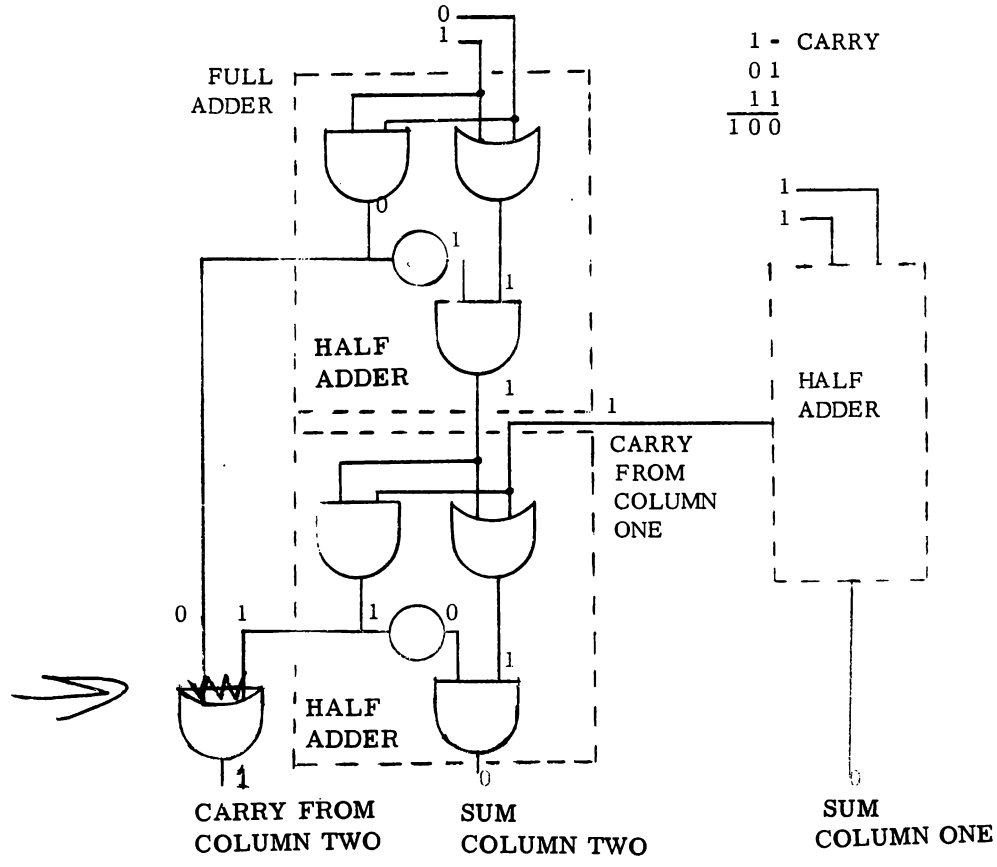


FIGURE 49
The Full-Adder

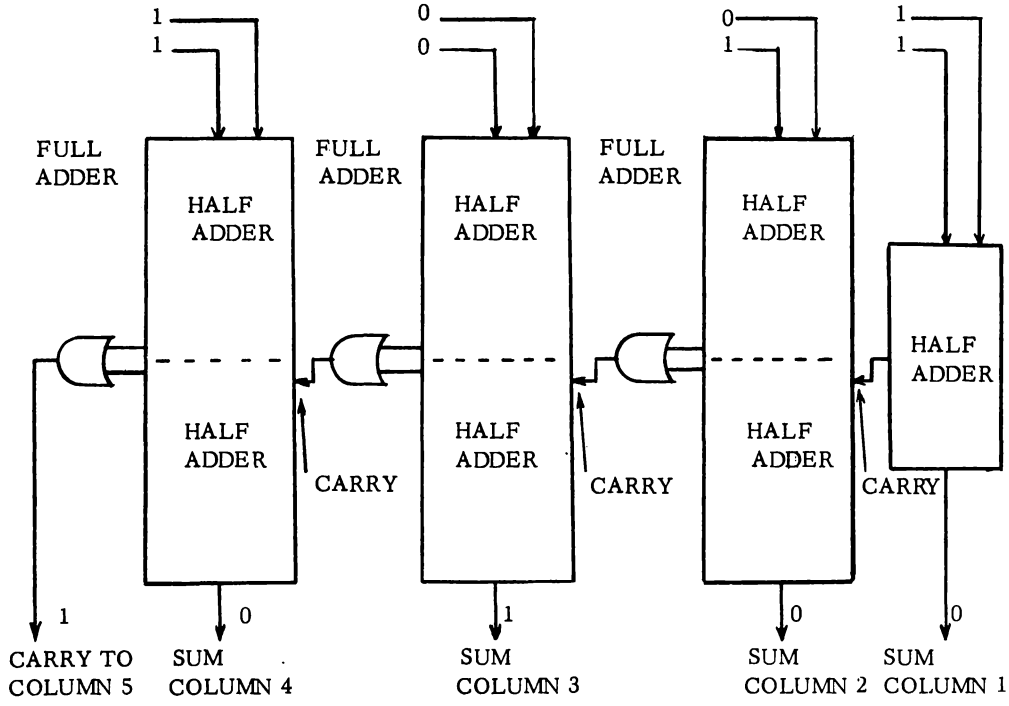


FIGURE 50
The Adders (Example: 1001 + 1011)

LARGE-SCALE STORAGE AND MEMORY

REQUIREMENTS OF MEMORY ELEMENT

The brief description of the memory element given in Chapter 1 stated that it consisted of a large number of storage locations, each having its own separate address and each capable of storing one computer-length word.

Without a memory such as this, the automatically sequenced computer could not perform a long string of arithmetic computations and logical decisions without stopping. It must have some place to keep all the input information it has to work with, the intermediate results that will be used in later computations, and the final results that have to be fed out when the program calls for outputs. In a stored-program computer the memory must also provide enough storage registers to hold all the instructions of the program.

The first requirement of the memory element, then, is size. It must contain enough storage registers to hold all the data and all the instructions of the program. In a very large computer, this may not be feasible and auxiliary storage space may be provided outside the main memory element. In this case, the memory must be sufficiently large to hold information and instructions enough to keep the computer running for a reasonable length of time.

Another important requirement of the memory element is the speed with which numbers can be put in or taken out. This is known as "access time" and is defined as the time interval between the instant information is requested and the instant it becomes available. In memory systems using the "random access" method, the time interval will be the same for the addressing of any location of the memory medium. The "access time" largely controls the speed of operation since many operations can be performed faster than the numbers can be obtained to work with. So, a fast-access memory is required and, once obtained, another reason for storing the program instructions in memory becomes apparent. For maximum operating speed, the instructions must be made available just as fast as the numbers to be operated upon, so the logical place to keep them is with the data numbers.

Part of the access time (sometimes called the memory cycle) must be used to translate the address (also in number form) and set up electrical connections to the desired storage register, in order to write in or read out a number. Writing or storing is the process of putting a number into a storage location; reading is the process of taking it out.

Translating the address in the address selection circuits rarely takes long, but actually reaching the proper storage register, when some types of storage devices are used, may require much time. The effect is as if the registers were seats on a merry-go-round or cars on a roller coaster and it were necessary to wait for the desired one to come by. Thus the counter may be forced to wait for the information it is to work with. Certain techniques in preparing programs can be used to cut this access time to a minimum, but these techniques often make the problem of writing the program very complex.

A better solution is to use fast-access storage devices for the main memory and use the slower devices as auxiliary storage facilities outside the computer proper, reached through input-output circuits. ("Memory" usually means the main storage element inside the computer.) Then, large groups of numbers at a time can be sent back and forth, as required, and stored in consecutive storage registers. Sometimes the computer can continue its computations during the transfer. Instead of having to locate individual registers in the auxiliary storage, the access is made to large blocks of registers.

Only the main memory is used for all operations going on inside the computer. When

the memory fills up with intermediate results, program instructions send a large block of them out to the auxiliary storage and may bring back in some fresh data or even additional program instructions, as required.

Of the magnetic storage devices (cores, tapes, and drums), the cores offer the easiest and by far the fastest access to any storage location. Magnetic cores are the most satisfactory storage devices for use in the internal computer memory, since all registers are equally accessible. The other types of storage devices generally require the computer to wait for the transfer of information, but they offer such other advantages as low cost, fast serial operation, or easy changing of the stored information by an operator.

Magnetic storage takes two principal forms: One stores the individual bits in separate cores; the other stores each bit by magnetizing a separate, tiny spot of a magnetic material coated on the surface of a plastic tape or a metal drum. In both forms of storage, the magnetic field that is left (remanent flux) after writing the information indicates by its direction (polarity) whether a "1" or a "0" is stored.

The magnetic material coated on the surfaces of tapes or drums acts like a permanent magnet, the direction of whose field can be reversed by applying a second magnetizing force of sufficient strength. This external force is usually a temporary magnetic field about a coil through which a pulse of current is passed.

A coating of such a material on a surface that is relatively flat does not form a closed magnetic circuit for small fields (as the closed ring of a core does), so separate areas of the surface can be magnetized in opposite polarities without interfering with each other, as long as there is sufficient distance between them. If the applied magnetizing force is kept in a very small field, only a correspondingly small spot of the tape or drum coating is magnetized and more bits can be stored on a surface of given size. The surface is moved past the stationary coil at a constant speed, and writing and reading are done with the surface in motion. This is illustrated in Figure 51. Long lengths of magnetic tape are wound on compact reels and pulled past the coils used for reading and writing (called magnetic heads). A magnetic drum revolves on its axis, passing its coated cylindrical surface under fixed heads.

When the small magnetized spots representing stored bits of information are passed again under the head, each tiny magnetic field enters and travels quickly around the magnetic circuit of the core, inducing a voltage pulse cycle into the head coil. The bit is identified (by reading circuits) as a "1" or a "0". The magnetized spots are unchanted by reading, so this is nondestructive readout; that is, the stored information remains on the tape or drum.

Any stored bit can be changed from "1" to "0" or from "0" to "1" simply by writing over it. It is also possible to remove all stored information from a tape or drum

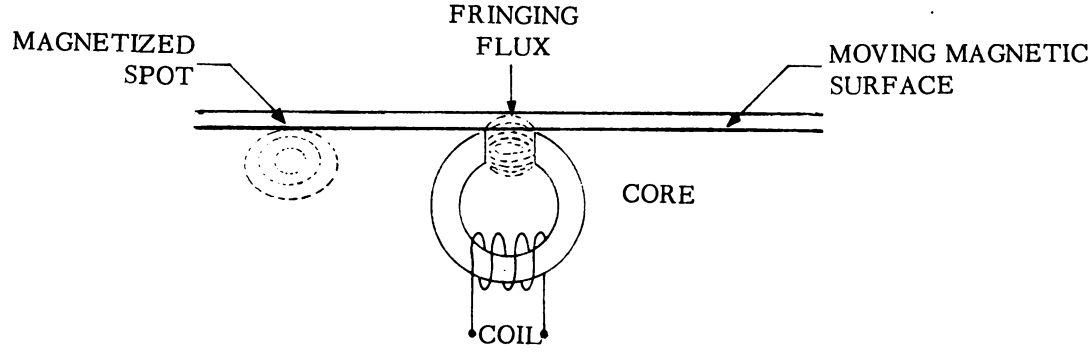


FIGURE 51
Magnetic Head

surface by "erasing", leaving a blank, unmagnetized surface.

Erasing of drums is not usually necessary to change the stored information, which can be simply written over (if it can be located easily), but is used to rid the surface of noise caused by stray magnetic flux picked up over a period of time. On tapes, where a single word or bit is difficult to locate, erasing is used to wipe out old information, a complete block at a time, to make way for new data.

I/O devices are described primarily in terms of the volume they can hold (Capacity) and the time required by the device to locate a particular address (Access time). In addition to these two characteristics, an I/O device is often discussed in terms of the maximum number of characters that the device transfers in a second (Transfer Rate); data (Mode); and the way in which the device transfers self-checking ability of the device (Validity).

The medium upon which the data are actually stored has several discernable characteristics. Among these are the ability for a medium to be re-used (Eraseability); the capacity of the medium to resist wear (Durability); the time required for a drum or disc to complete one revolution (Cycle Tine); the capacity of the medium to retain the stored data when the power source is lost or turned off (Non-volatility); the number of bits that can be stored per inch (Density); and the manner in which data are stored on the medium (Magnetic or Non-magnetic).

The CAPACITY of an I/O device is a measure of the maximum number of bits, or collection of bits, that a particular device can contain at a given time. As such, it is a measure of the amount of information available to the system from a given device.

PARALLEL AND SERIAL DATA TRANSFER

As it was mentioned in Chapter 4, data is transferred from or to an I/O device in one of two modes. A serial transfer, or transfer in a serial mode, occurs when data (generally in the form of binary digits or bits) are read, or written, one after another in a time sequence. That is, a serial transfer occurs when each bit of information is handled singly and successively. For example, if a computer word contains eight characters and each character is made up of six bits, a serial transfer of the data in that computer word would require forty-eight separate distinct read or write operations. That is, each bit of the computer word would be transferred individually and independently.

A parallel transfer, or transfer in a parallel mode, occurs when data are transferred simultaneously. Thus, a computer word containing forty-eight bits would be transferred in one read or write operation.

Data transfer is often a combination of the two modes. For example, if the data to be transferred are in a computer word which contains eight six-bit characters, the bits which make up the six-bit characters could be transferred simultaneously (parallel mode) and the characters transferred one after another (serial mode). Transferring data in a parallel mode is much faster than transferring data in the serial mode; however, the cost for circuitry to provide a parallel transfer is considerably higher than the cost for the circuitry necessary in a serial transfer of data. The decision to provide either parallel or serial transfer in an I/O device is made by the equipment manufacturer and is determined, in great part, by the existing requirements for speed and cost.

INTERNAL/EXTERNAL STORAGE

It is common practice in large scale computer-based systems to use several types of storage. These provide an area of working storage featuring immediate or relatively fast access time to a moderate number of words; a back-up or secondary I/O storage having a greater capacity but slightly slower access time than the working storage area; and a bulk storage area providing a virtually unlimited storage capacity with reasonably slow access times. Storage is often spoken of as being Internal Storage or External Storage.

Ideally, this means that the stored data are either contained entirely within the computer and controlled by the computer (internal), or that the data are completely without the computer and not directly controlled by the computer (external). However, computer storage requirements cannot be adequately described by the division of storage into internal and external. In an attempt to provide an adequate description of storage requirement, storage is frequently classified according to usage as Primary or Secondary storage levels.

PRIMARY STORAGE

Primary storage for a computer is that storage which features immediate or fast access to a moderate-to-large volume of data. If the computer system has magnetic core or thin film storage as well as magnetic disc, magnetic drum, or magnetic tape storage, the magnetic core and/or thin film storage is considered to be the primary storage for the system. If the system has only magnetic discs, magnetic drums, and magnetic tapes, then the magnetic disc and magnetic drums are considered to be the primary storage. Once the primary storage is determined, the remaining devices form the secondary storage level. Basically, and in a somewhat simplified form, primary storage holds the operating program and its associated data and is generally the fastest storage available which can transfer data directly to and from the Central Processing Unit. Secondary storage supports primary storage by providing a reservoir of data, programs, sub-routines, etc., which can be made available to the CPU by transferring them from secondary storage to primary storage.

SECONDARY STORAGE

Secondary storage forms the auxiliary or support area of storage for a computer system. Secondary storage will have, relative to primary storage, a larger capacity, slower access time, and cheaper cost per character storage. That is, secondary storage provides moderate to low cost per character storage for large amounts of data which do not have to be accessed quickly. For this reason, secondary storage is often used to store large data files, programs, libraries, directories, etc. Data stored in secondary storage are in machine-acceptable format and can be incorporated into the operating program or system with a minimum of delay. Magnetic tapes, magnetic cards, drums, and discs are often used as components of secondary storage.

Data stored on punched cards and punched on paper tape are treated somewhat differently than data stored on magnetic tape, core, drum, or disc. For this reason, they are sometimes considered as a third, or bulk, level of storage. Data stored on these devices are not directly available to the control element of the computer. This class, or level, of storage media provides for the storage of vast quantities of data at a very low cost per character. However, due to the nature of the media, access time is very slow.

Most data to be processed through a computer are, at one time or another, placed on punched cards. After flow-charting and coding a problem, you reproduce the coding onto punched cards, which then becomes your program. The cards can be loaded directly into memory and executed, or they can be first transferred onto magnetic tape and then transferred into memory. In any event, the cards represent another means of communication between you and the computer.

MAGNETIC TAPE

Magnetic storage devices can have extremely fast random access times or much slower sequential access times. Magnetic storage media are used at both the primary and secondary levels of storage and are available in the form of cards, tapes, drums, discs, cores and thin film. Magnetic tape is one of the most popular mediums on which to save information and is the I/O device we will discuss in some detail.

Multiple reel storage affords an almost unlimited storage capacity and provides a safe, permanent storage medium which can be

erased and re-used thousands of times. A ten-inch reel of magnetic tape can hold between six million and thirty million bits of information depending upon the number of tracks and the density at which the characters or bits are packed. Magnetic tapes are generally 1200 to 2500 feet long and may have densities of 200 to 1500 bits per inch (bpi) along each track. Under program control, the tape can be positioned forward or backward. It can be rewound to the start of the tape and the tape drive can be prepared for tape loading or unloading.

Data are stored on magnetic tape by magnetizing spots on the tape surface. The tape surface is generally divided into several tracks (Figure 52). Each track is considered to be one bit wide and occupies the length of the tape. A collection of the "n" bits contained in the "n" tracks across the width of the tape (Figure 53) constitutes a character. The number of tracks on a tape may vary between six and thirteen, although the majority of tapes now produced contain seven or ten tracks.

A seven track tape contains a six-bit character along with an associated control bit across its width. The seventh bit is often used as a validity or error checking bit. When used in this manner, the seventh bit is known as a longitudinal parity bit. The parity bit indicates that the sum of the associated six information bits is either an odd or even number (Figure 53). The parity bit is recorded during the write operation and is used to check the accuracy of a data transfer during a read operation.

Ten track tape can contain a six-bit character and four associated control and error checking bits (Figure 55) or two four-bit BCD characters and two control and/or error-checking bits (Figure 54), or other appropriate combinations of x-bit characters and control bits. The use and function of the control bits is determined by the manufacturer of the tape transport.

A magnetic tape which is used for data storage will contain two physical marks on the tape (Figure 56). These two marks are known as the Beginning-of-Tape mark (BOT) or load point, and the End-of-Tape mark (EOT). The area bounded by these two marks is reserved for data storage.

Within the area defined by the BOT and EOT, data are recorded in logical units which are known as RECORDS. Records may or may not have a fixed length. An arbitrary record legnth of size may be specified by the device manufacturer or it may be fixed by the program/system designers. If a record has no specified maximum length, the determination of record length depends upon the data being handled. In addition to containing a set of related program data, a record can also contain a label which is used to discriminate between groups of records, e.g., a word, a mark, or a character. Each record may also have a lateral parity character associated with it (Figure 57).

Lateral parity exists when the bits on each track of the record are summed and an additional bit is added to the track to indicate that the sum was either an odd or an even number. Each track is treated

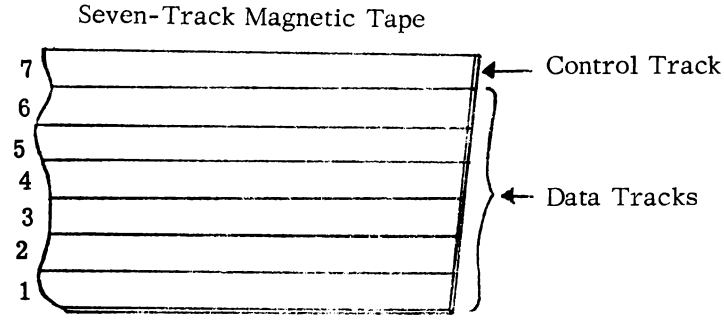


FIGURE 52
Seven Track Tape--Even Parity

Seven-Track Tape--Even Parity

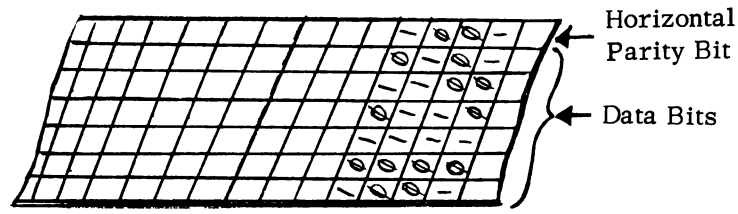


FIGURE 53

Ten-Track Magnetic Tape

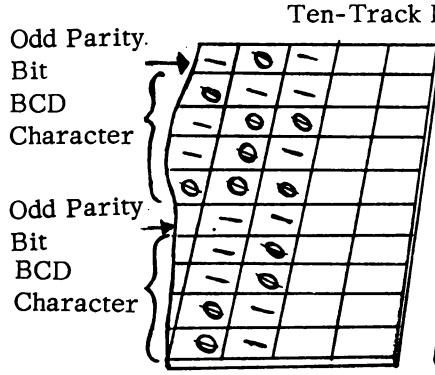


FIGURE 54

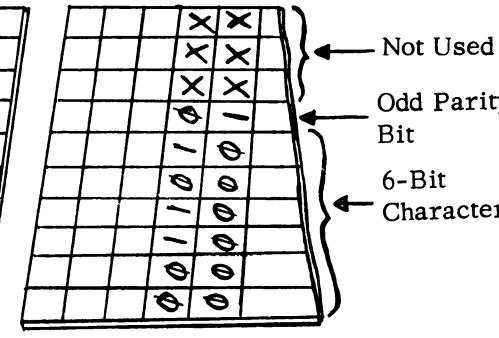


FIGURE 55

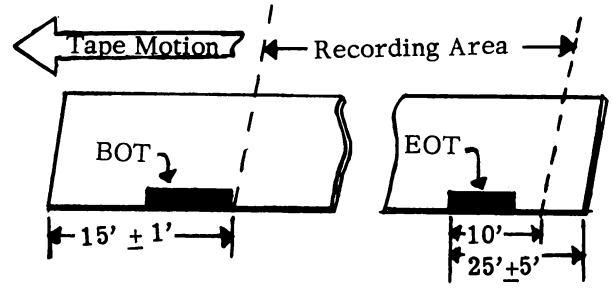


FIGURE 56

EVEN Horizontal Parity & ODD Vertical Parity

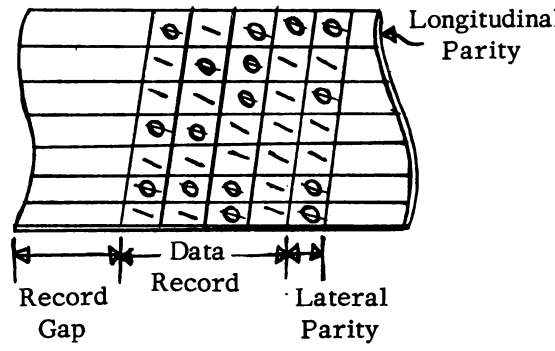


FIGURE 57

individually and one additional bit is added to the record following the data bit of the record (Figure 57). Lateral parity is determined during the writing operation and provides for an additional check on the accuracy of data transfer.

Each record is separated from the records preceding or following it by an area of blank or data free tape referred to as a record gap or an End-of-record gap (EOR). The record gap is a blank spot on the magnetic tape over which the read-write head will be positioned following a "read record" or before a "write record" operation. Effectively, the length of a record gap is determined by the amount of tape which passes under the read-write head as the tape motion goes from full speed forward to a full stop and back to full speed forward.

Records are separated by record gaps and grouped together to form FILES (Figure 58). A file is a collection of logically related records. Several files may be on one tape or one file may require several reels of tape. The length of a file is generally not fixed. Files are separated by end-of-file marks (EOF). An EOF mark is usually special manufacturer-determined characters or symbols. As files do not have a fixed length, the EOF mark provides a means of determining when or where a particular file ends on the tape (Figure 58).

Data are generally read from or written onto tape in terms of records. That is, the program-directed control element of the computer will cause the tape to move forward until the next EOR gap is under the read-write head in a read operation. The read head transfers the data that passes under it to a designated area in core memory. If the data are to be transferred from core to tape, the program-directed control element of the computer will cause an EOR gap to be generated after the data has been transferred to tape.

MAGNETIC DRUMS

Though tapes are valuable for storing large amounts of information, when it is essential to write and read information at frequent intervals and in random order, magnetic drums offer much faster access times, commonly ranging from 10 to 40 milliseconds. Because the information is stored on the surface of a cylinder revolving under fixed magnetic heads, the drum provides a form of cyclic storage (once written a word comes back under the heads on every revolution).

As the drum rotates, the area in which a single fixed head can write or read is only a very narrow strip (called a track or channel) running around the circumference of the drum. Information can be stored in serial form simply by sending serial words to the single head while the drum revolves (translating "0's" and "1's" to current pulses of the proper polarities). The bits of each word are then stored as a sequence of magnetized spots along the single channel running around the drum.

Another common storage method is parallel storage, shown in Figure 59. To store a 5-bit word by this process, five heads are lined up side by side, each writing in a separate channel. The translated current pulses representing the bits of the word are sent in parallel form to the heads and the bits are written simultaneously. Now, the bits are stored as a row of magnetized spots in adjacent channels. So, in this method, the registers are strips of drum surface running toward the ends of the drum and including as many channels as there are bits in the computer word. In the example of Figure 59, a register stretches across five channels.

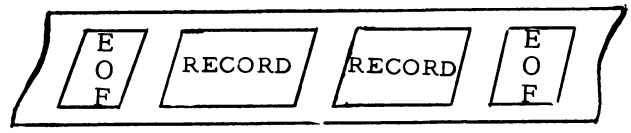


FIGURE 58

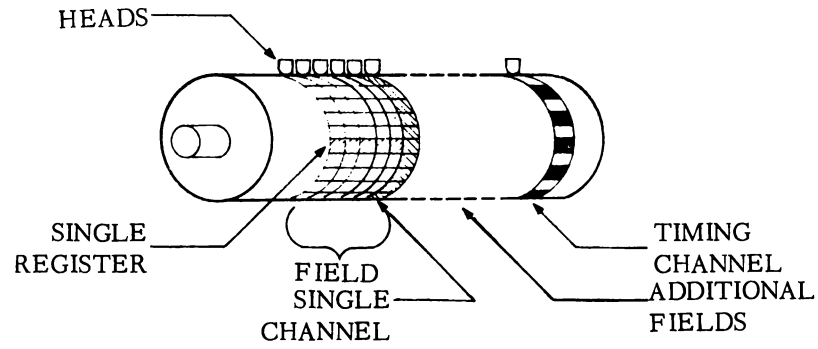


FIGURE 59
Storage of Magnetic Drum

The band of registers extending completely around the drum is called a field.

Drum speeds, sizes, and capacities vary greatly. The maximum access time is a function of drum speed and drum diameter which causes the information to pass the heads at a given rate. A typical drum used with a Weapon Support System computer may hold six fields of 2,048 registers each, for a total storage capacity of 12,288 words.

Locating a given register or group of registers on the rotating drum or read or write information requires some means of keeping track of the drum position. One common method uses a special timing channel in which is written either a series of "1's" or a regularly repeated combination of "1's" and "0's". These bits are read by the timing channel head and used to synchronize the access circuitry with the drum rotation and to locate registers by a cycling count. A special combination of "1's" and "0's" at one point on the track can be used as an index mark to tell the circuits that a new revolution of the drum is beginning.

MAGNETIC DISC

The magnetic disc memory device consists of many discs fastened to a shaft. They are spaced to allow arms, each with a read-write head, to pass between the discs. The discs are coated with a material that can be magnetized rather than changing the physical shape of the disc as is done when recording music on a phonograph record. The shaft which is through the center of the disc is usually vertical and is rotated at a constant high speed. This type of memory device usually has a greater capacity, but a much longer access time, than the magnetic drum and is considered an external memory device.

MAGNETIC CORE

Coincident current magnetization is the term used when two current-carrying wires intersect at an element of the static storage and magnetize that element. This is the method used to magnetize the ferro-magnetic toroids that form a magnetic core storage device.

Each core in a magnetic core storage device is capable of accepting and retaining magnetic polarity in one direction or another. This means that a core can be magnetized in a positive or negative direction. A core, then, is a bi-stable device and can be used to record binary information in the ratio of one bit of information to one core.

Every core occupies the intersection of three wires. A current of some magnitude "i" is required to magnetize a core. If two wires intersect a core and each carries a current "i/2" to the core, the sum of the two currents equals "i" and the core becomes magnetized. If only one of the two wires is carrying a current, the core will not be magnetized. The third wire is called a sense wire. The sense wire interrogates a core to see whether or not that core is magnetized.

If a computer word consists of 12-bits, a core storage device (core memory)

would have to have 12 planes of cores arranged in the 64x64 matrix. The computer word would consist of 1 core from each plane along the vertical axis. Each set of 12 cores would occupy the same relative position throughout the 12 planes and would have a unique address. Figure 60 illustrates a 12 bit word with appropriate current-carrying wires and a sense wire. This arrangement allows each core to be set to a magnetic state independently of the others. The sense wire can sense every bit (core) of the word and determine the number of bits (cores) that are magnetized (binary 1) and the relative location of each core that was set.

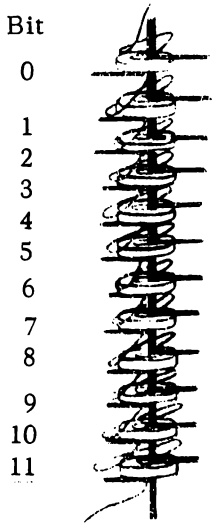


FIGURE 60

Cores which have been set to a magnetic state can be re-set (returned to zero) by changing the direction of the current in the two current-carrying wires which intersect the core (Figure 61). Cores can be set, re-set, or sensed at the rate of one operation per cycle or one operation per command-generator pulse.

THIN FILM STORAGE

Thin film storage is another form of static storage. At the present time, thin film storage is being used in conjunction with ferro-magnetic core memories. The thin film memory provides a small to moderate storage capacity which features very low random access time.

Thin film storage is produced by depositing tiny particles of a magnetic medium onto an insulating base such as glass, gold, silver, or aluminum. Each particle is only a few millionths of an inch in diameter and the entire substrate is frequently less than 5000 angstroms thick. (1 angstrom = one one-hundred-millionth of a centimeter.) The medium which is deposited onto the base is generally a nickle-iron alloy which can be magnetized. Each particle of the substrate can be compared to a magnetic dipole, or flip-flop. That is, each particle can assume an "on" or "off" state. The particles can be deposited with densities varying between 200 and 1000 bits per square inch. After the substrate has been deposited onto the base, the base is sandwiched between two read-write sense circuits to form an element of the thin film storage device.

The thin film storage element records data in much the same way that a ferro-magnetic core memory does. The read-write sense circuits which encase the thin film substrate cause the previously magnetized particles to be magnetized in one of two directions. One direction represents a binary 1, the other direction records a binary 0. Thin film memory is read by a sensing conductor which records the binary configuration

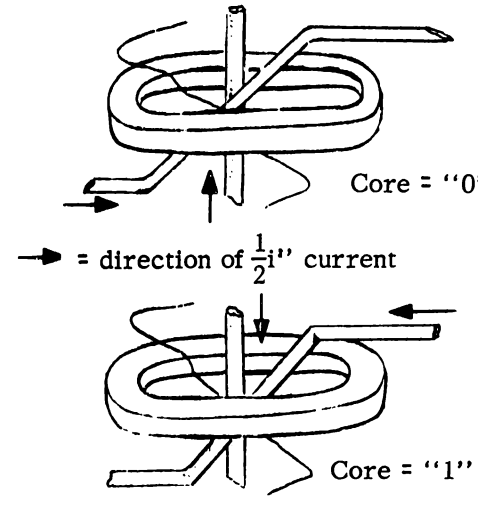


FIGURE 61

of the particles. Read-out can be either destructive or non-destructive.

PUNCHED CARDS

Punched cards are probably the most common and well-known form of all data storage media. Punched card storage offers unlimited storage capacity at an extremely low cost per character. The data stored on a punched card as a series of punches can occur in several different forms, e.g., 12-bit Hollerith, 6-bit per character format, and binary. In many of these representations, the character corresponding to the punched code can be printed across the top of the card. This feature permits data stored on cards to be easily identified, maintained, and modified.

As has been discussed previously, cards generally form the original machine-oriented representation of program. That is, a program or system will first occur on cards and will, at some later date, be stored on some faster medium for operational use. In most cases, the original program decks will be saved and maintained to provide redundancy or back-up protection.

The most common type of punched card is the IBM (International Business Machines) 80-column card (Figure 62). The card is divided into 80-vertical columns, each one capable of storing one or two characters. Each vertical column is divided into 12 horizontal rows. Information is represented by the presence of a rectangular hole punched according to one of several schemes. (See the chapter on Information Representation.) A total of 960 distinct positions are available on each 80-column card.

Another type of punched card is the Remington Rand 90-column card (Figure 63). The Remington Rand card is divided horizontally into two halfs. Each half is divided into 45 vertical columns. Each vertical column (half a card) can contain one

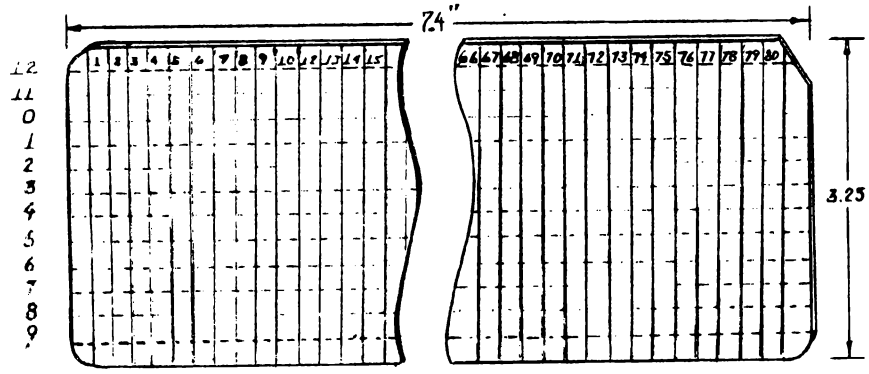


FIGURE 62

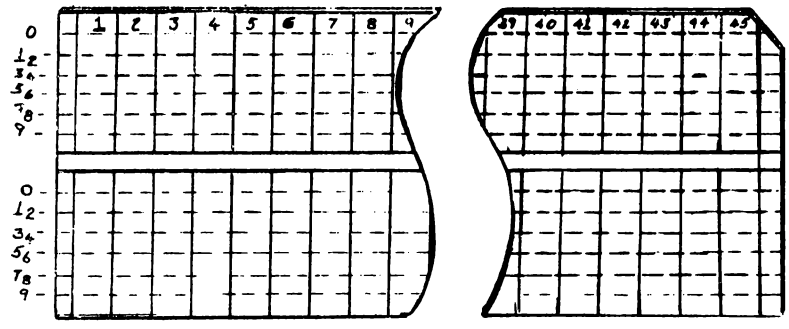


FIGURE 63

character. Information is represented by combinations of one, two, or three round punched holes. A total of 540 bits of information can be represented in a binary fashion on a Remington Rand card.

Punched cards are read by one of three common techniques. These techniques are the brush or electric contact method, mechanical probe method, or the photo-electric method.

The electric contact method passes a card between a contact roller and a set of brushes. The brushes are kept from touching the contact roller by the card until a punched hole passes under the brushes. When this happens, the brush drops into the hole and touches the contact roller. This completes an electric circuit and generates an electric impulse.

In the mechanical probe method, the card is moved under a set of probes which rest against the card. As a punched hole occurs, the probe drops through the hole and completes an electric circuit. Both the electric contact method and the mechanical probe method are mechanical in nature and are subject to mechanical malfunctions as well as slow reading speeds.

The photo-electric card reader operates in much the same way as the photo-electric paper tape reader. The card is moved past a beam of light and acts as an opaque substance until a hole passes under the light source. The light beam passes through the hole and activates a photo cell. This in turn generates an electric pulse or signal which is transmitted to a storage device.

Mechanical punched card readers generally read a card a row at a time. Reading speeds for this type of device range between 100 and 900 cards per minute. Photo-electric card readers usually read cards a column at a time at speeds up to 2000 cards per minute.

Cards are produced by manually operated equipment or by computer controlled card punches. These card punches can produce punched cards at a rate of 250 to 300 cards per minute.

BUIC III EQUIPMENT CONFIGURATION

The commercial name for the computer equipment used by the BUIC system is Burroughs D-825 Modular Processor. The D-825 is a solid-state, internally stored program computer that is based on the principle of total modularity. Total modularity allows the equipment to be freely organized and expanded by the use of combinations of standard computer modules, memory modules, and input/output nodules.

The D-825 is organized into a specific configuration for the BUIC III system. The central data processing configuration has been given the name AN/GYK-10 and is comprised of the following:

- 2 computer modules
- 8 core memory modules
- 4 input/output modules
- 2 message processors
- 3 magnetic storage drums

The full configuration of equipment used for the BUIC system is given the name AN/GSA-51A. It includes the AN/GYK-10 equipment and the following terminal devices:

- 10 or 11 data display consoles
- 4 magnetic tape drive units
- 1 tape drive control unit
- 1 status display console
- 1 Flexowriter
- 1 on-line printer
- 1 card reader
- 1 simulator group

DESCRIPTION OF BUIC III EQUIPMENT

COMPUTER MODULES

The AN/GSA-51A has two computer modules, each being a central processing element for AN/GSA-51A operations. The main function of the computer module is to decode program instructions and to provide the control and logic circuitry and the arithmetic registers necessary for performing the decoded instructions.

Each computer module performs its functions independently of the other computer module. However, each can be programmed to interrupt or begin operation of the other one. Because the computer modules operate independently, two separate programs can be operating simultaneously on the AN/GSA-51A.

There are three functional areas within each computer module. The ARITHMETIC UNIT contains registers and circuitry for performing operations specified by the instructions. The CONTROL UNIT contains circuitry for controlling the operation of instructions and program timing. The SET OF THIN FILM REGISTERS contains 128 registers which are used for data storage and program control.

Each computer module also has an external control panel which is used for performing manual operations necessary to start a program, for performing maintenance operations, and for debugging programs.

CORE MEMORY MODULES

The eight core memory modules are used to hold program instructions and data. Each memory module contains 4096_{10} (or 10000_8) ferrite core locations. Each location has 51 cores (bits) for storing an instruction or data word. Forty-eight of these bits are information bits, one bit is a parity bit, and the remaining two bits are spares. Throughout this document, reference will be made to "48-bit core memory words" because the programmer is concerned only with the 48 information bits in the core memory location and also because this is common terminology among AN/GSA-51A programmers.

All core memory locations may be accessed randomly. Information is transferred between the memory modules and the computer modules and between the memory modules and the input/output modules. Other pieces of equipment which require transfer of information to or from core memory must access the information through the use of the input/output module. Transfer of data in and out of core memory requires 4.33 microseconds.

The eight core memory modules can be thought of as one large memory area. Each memory word is given its own absolute address which distinguishes its location from all other memory locations. The locations are given addresses in consecutive order with octal numbers. They begin with 0 and end with 77777_8. The memory modules themselves are numbered beginning with 1.

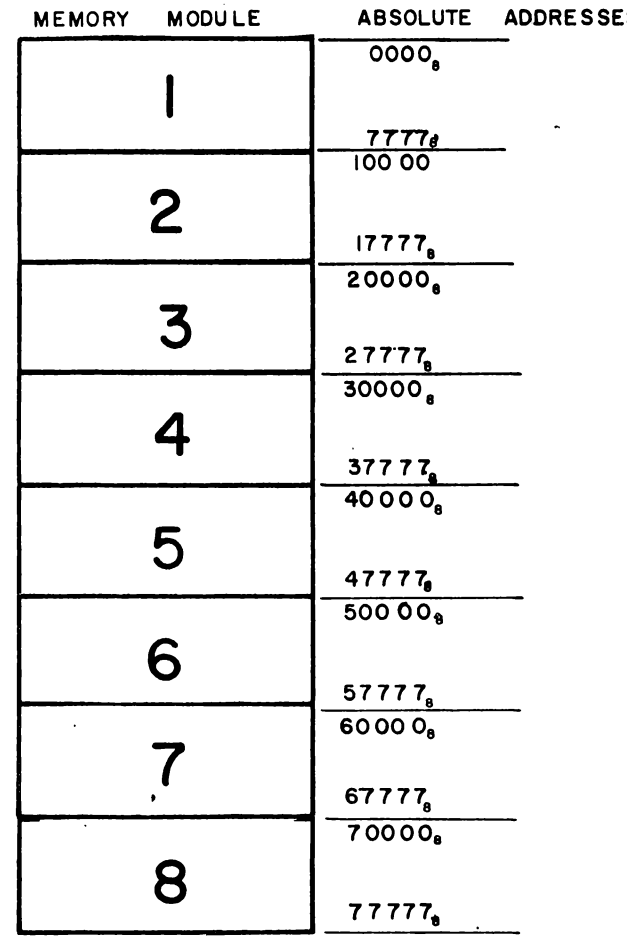


FIGURE 64

INPUT/OUTPUT MODULES

The four input/output (I/O) modules provide the control circuitry necessary for data transfer between the core memory modules and the I/O terminal devices. An input/output operation is defined and initiated

by computer module action but then proceeds independently under the control of one of the I/O modules. All four I/O modules can be handling separate I/O operations simultaneously.

Compatibility and connection between the I/O modules and the terminal devices is provided by an I/O EXCHANGE. The I/O exchange permits data flow between an I/O module and the terminal device being used. It consists physically of circuitry found in the I/O modules and in the terminal devices.

MESSAGE PROCESSORS

The message processors have three basic functions:

1. To provide a temporary storage for messages accumulated from radar sites, other BUIC NCCs, SAGE DCs, and other related facilities.

2. To change the format of information sent from other facilities to 48-bit words usable by the BUIC system.

3. To change the format of information sent to other facilities from the 48-bit word format to the format used by them.

MAGNETIC STORAGE DRUMS

Each of the first two magnetic storage drums provides $65,536_{10}$ words of storage. Some of the words ($39,936_{10}$) on each of the two drums may be used for bulk storage of program instructions and data. The remaining $25,600_{10}$ on these two drums are used to provide automatic readout and transfer of display data to the data display console. The third magnetic drum is used entirely for bulk storage.

When a display is prepared by the computer and memory modules, it is sent to the magnetic storage drums through the I/O modules. The drums then automatically begin sending the display to the data display consoles and will continue to send it until the display information is erased from the drums.

DATA DISPLAY CONSOLES

The data display consoles are used to display radar and tracking information on cathode-ray type scopes and also to input information into the AN/GYK-10 to be processed by the BUIC programs. The console operators input such information by pressing buttons on keyboard panels on the data display consoles. They may also use a light gun on the main scope to aid in sending positional information to the computer.

STATUS DISPLAY CONSOLE

The status display console provides a means for monitoring the operational status of all AN/GSA-51A equipment, provides a way of applying and removing power for itself and for the rest of the AN/GSA-51A, and assists in the repair of faulty modules.

FLEXOWRITER

The Flexowriter is an electric typewriter and a paper tape punch and reader combined into one unit. It is used for the exchange of information between the operator and the operating program. The paper tape punch and reader are run at the option of the operator. However, at all times, the electric typewriter provides a hard-copy record of the exchange of information regardless of whether the exchange was made through the typewriter or through the paper tape punch and reader.

PUNCH CARD READER

The punch card reader reads 12-row, 80-column punch cards into central processing modules of the AN/GSA-51A. It is capable of reading punch cards at a maximum rate of 200 cards per minute. Information being read is sensed by photoelectric cells at a read station. The infornation is transferred to the card reader control which translates the card codes into AN/GSA-51A character codes. The character codes are sent, one character at a time, to the I/O module that is controlling the input operation. The I/O module then sends the information to one of the memory modules so that it will

be available for use by the controlling program.

ON-LINE PRINTER

The on-line printer is used to print alphanumeric data directly from core memory. It prints a maximum of fifteen words (120 characters) per line at a maximum rate of 600 lines per minute.

MAGNETIC TAPE DRIVE UNITS

The magnetic tape drive units are used to process magnetic tapes which provide bulk storage for large quantities of program instructions and data. Some of the operations of the tape drive units are performed under the control of the tape drive control unit. Those operations which the tape drive units can perform without the use of the control unit are rewind, backspace, advance, load, and unload.

TAPE DRIVE CONTROL UNIT

The one magnetic tape drive control unit controls all four magnetic tape drives. Under its control, the tape drive units perform read, write, advance, backspace, rewind, and erase operations.

SIMULATOR GROUP

The simulator group is used to provide the central data processing equipment with manually composed messages that simulate inputs from two interceptor pilots to effect testing and personnel training.

FLOW OF INFORMATION WITHIN THE COMPUTER

The operation of the AN/GSA-51A is controlled by an internally stored program located in core memory. All data flow occurs between I/O modules and memory modules and between computer modules and memory modules. There is no direct data transfer between computer and I/O modules.

Information flows directly between computer modules and memory modules over data transfer buses. Each computer has its own bus, Computer BUS 1 for computer module one and Computer BUS 2 for computer module two. The information being transferred will either be program instructions going to the computer to be decoded and operated, or data words being fetched from memory for operands, or the results of an operation going to memory to be stored.

Information flows directly between I/O modules and memory modules over data transfer buses called I/O BUS A and I/O BUS B. I/O BUS A is time-shared by I/O modules one and two. I/O BUS B is time-shared by I/O modules three and four. If both I/O modules on one I/O BUS simultaneously request access to core memory, the I/O module with the lower number will be granted access first, unless the program specified otherwise.

All core memory modules are accessible by all computer and I/O buses. Therefore, core memory is totally shared. Conflicts will arise if more than one computer or I/O BUS simultaneously request access to the same memory module. A SWITCHING INTERLOCK is provided to resolve these conflicts by handling the requests in order according to priority levels inherent in each request. The inherent priorities are as follows (from highest to lowest):

a. I/O BUS A

b. I/O BUS

c. Computer BUS 2

d. Computer BUS 1

While a computer or I/O bus is accessing a particular memory module, all other computer and I/O buses are denied access to that memory module until the data transfer is complete. During this time, however, another memory module may be accessed by any other I/O or computer bus.

The switching interlock consists physically of a portion of the circuitry in each of the computer, memory, and I/O modules.

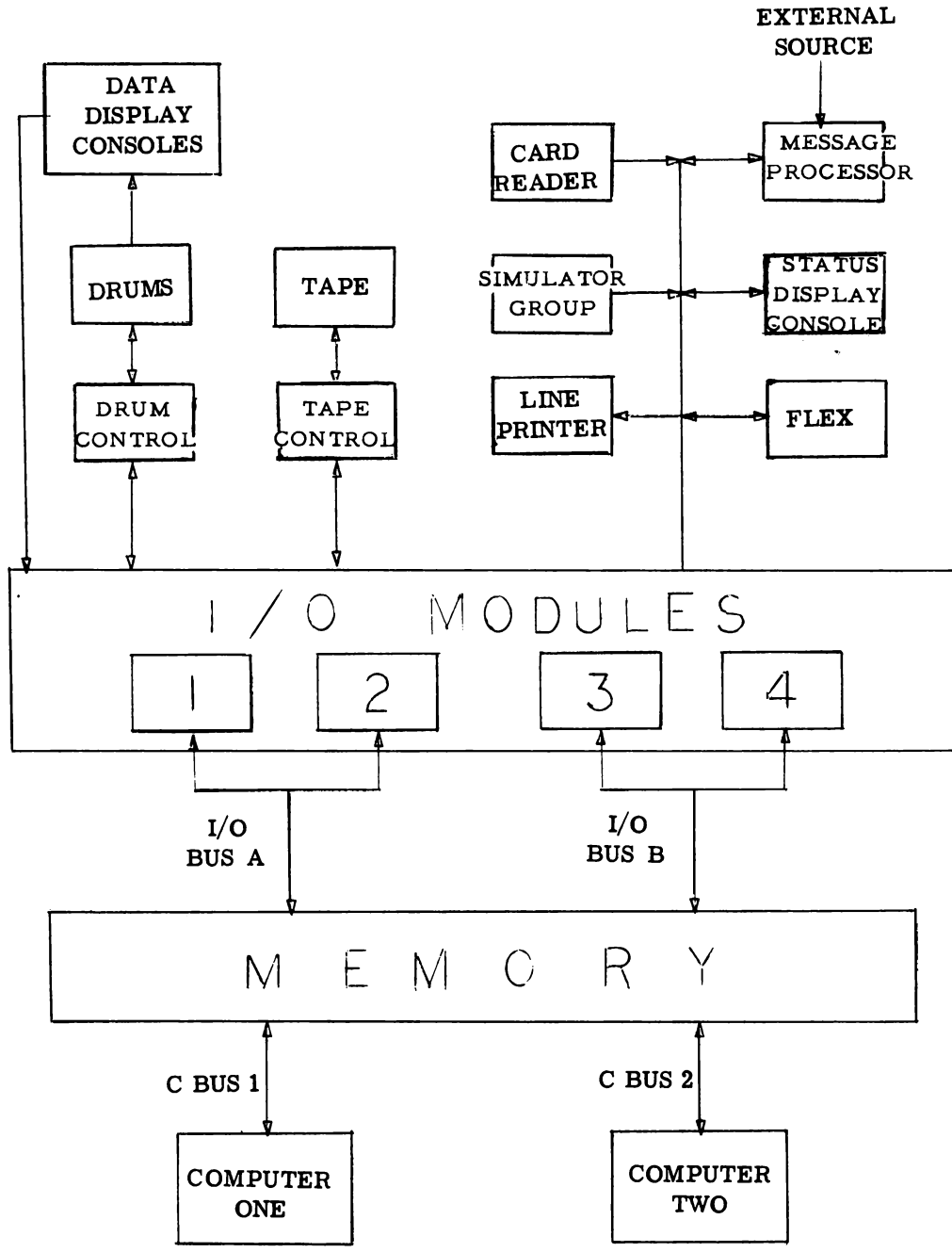


FIGURE 65
BUIC III Equipment Configuration

NOTES

NOTES

NOTES

SAVE A LIFE

If you observe an accident involving electrical shock,
DON'T JUST STAND THERE - DO SOMETHING!

RESCUE OF SHOCK VICTIM

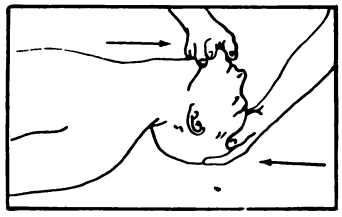
The victim of electrical shock is dependent upon you to give him prompt first aid. Observe these precautions:

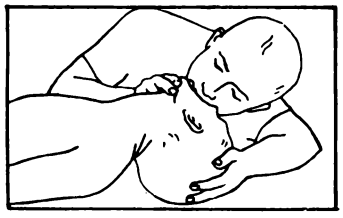
1. Shut off the high voltage.
2. If the high voltage cannot be turned off without delay, free the victim from the live conductor. REMEMBER:
   a. Protect yourself with dry insulating material.
   b. Use a dry board, your belt, dry clothing, or other non-conducting material to free the victim. When possible PUSH - DO NOT PULL the victim free of the high voltage source.
   c. DO NOT touch the victim with your bare hands until the high voltage circuit is broken.

FIRST AID

The two most likely results of electrical shock are: bodily injury from falling, and cessation of breathing. While doctors and pulmotors are being sent for, DO THESE THINGS:

1. Control bleeding by use of pressure or a tourniquet.
2. Begin IMMEDIATELY to use artificial respiration if the victim is not breathing or is breathing poorly:
   a. Turn the victim on his back.
   b. Clean the mouth, nose, and throat. (If they appear clean, start artificial respiration immediately. If foreign matter is present, wipe it away quickly with a cloth or your fingers).





   c. Place the victim's head in the "sword-swallowing" position. (Place the head as far back as possible so that the front of the neck is stretched).
   d. Hold the lower jaw up. (Insert your thumb between the victim's teeth at the midline - pull the lower jaw forcefully outward so that the lower teeth are further forward than the upper teeth. Hold the jaw in this position as long as the victim is unconscious).
   e. Close the victim's nose. (Compress the nose between your thumb and forefinger).
   f. Blow air into the victim's lungs. (Take a deep breath and cover the victim's open mouth with your open mouth, making the contact air-tight. Blow until the chest rises. If the chest does not rise when you blow, improve the position of the victim's air passageway, and blow more forcefully. Blow <u>forcefully</u> into <u>adults</u>, and <u>gently</u> into <u>children</u>.
   g. Let air out of the victim's lungs. (After the chest rises, quickly separate lip contact with the victim allowing him to exhale).
   h. Repeat steps f. and g. at the rate of 12 to 20 times per minute. Continue rhythmically without interruption until the victim starts breathing or is pronounced dead. (A smooth rhythm is desirable, but split-second timing is not essential).

DON'T JUST STAND THERE - DO SOMETHING!