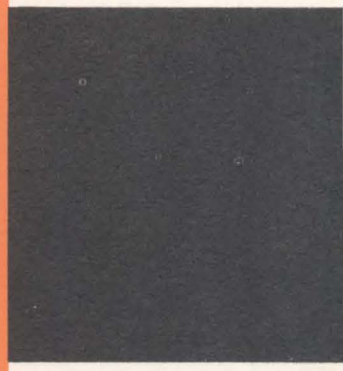


General Information Manual

Introduction to IBM Data Processing Systems

IBM

General Information Manual

Introduction to IBM Data Processing Systems

Preface

All IBM Data Processing Systems, regardless of size, type, or basic use, have certain common fundamental concepts and operational principles. This manual presents these concepts and principles as an aid in developing a basic knowledge of computers. The manual is designed for use in training programs where a basic knowledge of computers is the end objective or is a prerequisite to the detailed study of a particular IBM system.

Each section is organized to present a logical association of related concepts and operational principles. The sections may be used in a progressive sequence to develop a concept of the computer system, or they may be used independently as reference material. The subject matter has been generalized and refers to actual machines and systems as little as possible. Specific systems are mentioned only to illustrate a general principle, not to compare one system with another.

The appendix contains supplementary reference information which, because of its format or its logical use, is not related directly to any specific section of the manual.

Contents

INTRODUCTION TO IBM DATA PROCESSING SYSTEMS 5
The Data Processing System 12

DATA REPRESENTATION 17
Computer Data Representation 18
Computer Codes 20
Data Recording Media 23

STORAGE DEVICES 29
Core Storage 30
Magnetic Drum Storage 32
Magnetic Disk Storage 33
Storage and Data Processing Methods 34

CENTRAL PROCESSING UNIT (CPU) 36
Functional Units 37
Machine Cycles 38
Serial and Parallel Operation 40
Fixed and Variable Word Length 40

INPUT-OUTPUT DEVICES 41
Card Readers 43
Card Punches 44
Magnetic Tape Units — Input and Output 44
Paper Tape Reader 49
Paper Tape Punch 50
Printers 50
Cathode Ray Tube 52
Consoles 52
Data Buffering 52
Auxiliary Operation 54

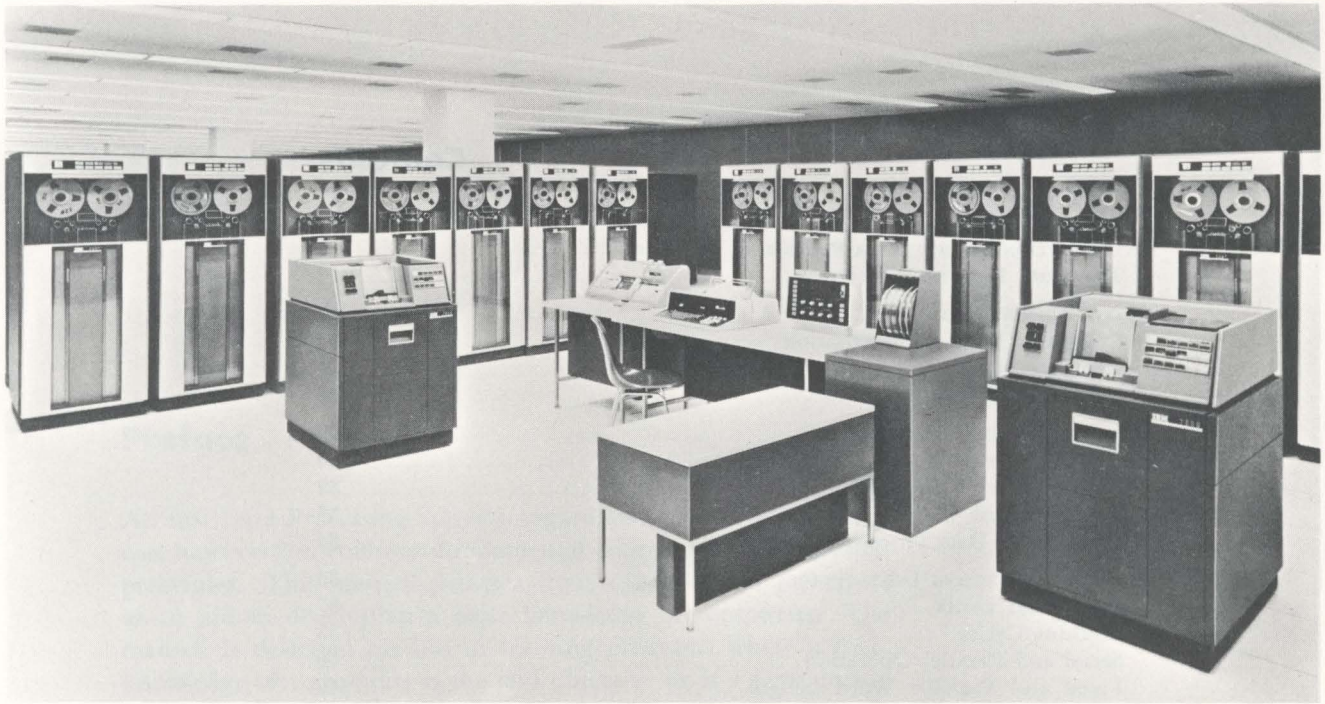
STORED PROGRAM CONCEPTS 56
Instructions 56
Developing a Program 57
Reading Data 64
Calculating 64
Logical Operations 65
Comparing 68
Instruction Modification 68
Address Modification 69
Indexing 70
Indirect Address 71

PROGRAMMING SYSTEMS 72
Machine Coding 73
The Programming System 73
Automatic Coding System 75
Macro-instructions 78
Program Compilers 80
Program Package 81
Fortran 82
Sort Programs 83
Utility Programs 84

PROCEDURE CONTROL 85
Systems Checks 86
Machine Checking 89

APPENDIX 90
Business Practices 90

INDEX 94



IBM 7070 Data Processing System

Introduction to IBM Data Processing Systems

TECHNOLOGICAL ADVANCE in data processing is fast moving and far reaching. What is in the future? No one really knows. The undiscovered ways in which data processing systems can probably be used seem almost boundless. With each new application, data processing systems have demonstrated still newer ways in which they can be used to help man enlarge his capabilities and advance civilization a little farther.

In the opinion of some scholars, data processing is not just one more new industry or innovation, but a giant step forward in man's utilization of science and knowledge as a means to progress. Ultimately, some say, the changes that may come in the wake of these developments will prove more momentous than those of the industrial revolution. Infinitely more is ahead than is in the past.

Data Processing

Data processing systems ordinarily consist of a combination of units including input, storage, processing, and output devices (Figure 1). They are designed to handle business or scientific data at electronic speeds with self-checking accuracy. The key element of these systems is the processing unit, a high-speed electronic computer.

These electronic data processing systems are a post-World War II innovation. Within two decades, they

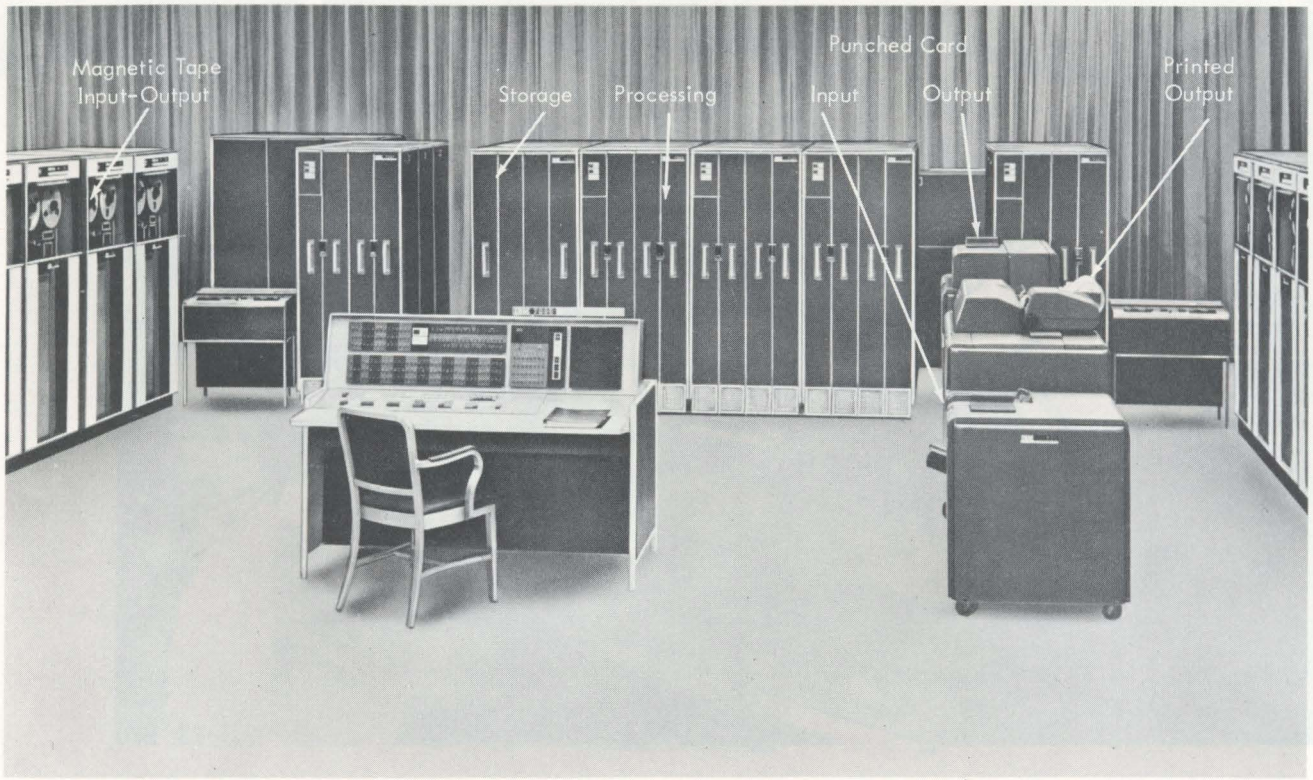


Figure 1. IBM 7090 Data Processing System

have progressed from experimental laboratory equipment to machines whose capabilities are exceeded only by the range of applications to which they can be put (Figure 2).

Machines are devised by men for a purpose. In the case of data processing machines, the purpose can be expressed simply: they offer man a means to increase his productivity.

They do this in two ways. First, they enable man to increase his output per hour and the quality of his output; this is true whether it be in research, production, problem solving, or the distribution of goods and services. Second, these machines increase productivity by encouraging careful and intelligent planning.

Data processing machines came into being primarily to meet the increasing need for information under increasingly complex conditions. High-speed data processing machines were not essential to the agricultural economy of a century ago. If they were, much greater effort would have gone into their development at a much earlier date. For while electronic techniques are new, the concept of automatic data processing is not; others perceived it more than a century ago.

As a manufacturing economy developed during the 19th century, it became clear that expanded markets would require mass production techniques. Machinery was introduced to increase productivity. It became possible to turn out more and more goods with less human effort. The work week shortened. Wages and profits went up. Benefits spread throughout the whole economy.

During the last quarter century, further changes have taken place. In many respects, they are as significant as the changeover from an agricultural to an industrial nation. Science has moved into the forefront of human activity. Research has grown to a multibillion dollar a year undertaking. New technology has provided a new impetus for corporate growth. Service industries have multiplied. Patterns of consumer spending have changed.

As these changes gained force, they manifested themselves in many ways. Informational needs greatly increased. Data assumed new importance. Clerical tasks multiplied. Paper-handling tasks appeared as if they would overwhelm all productive activities.

Today, more people are engaged in the handling, processing, and distribution of goods and services than are engaged in their production. One dollar out of every eight in wages and salaries in the United States now goes to a clerical worker. White-collar workers in manufacturing industries have increased by more

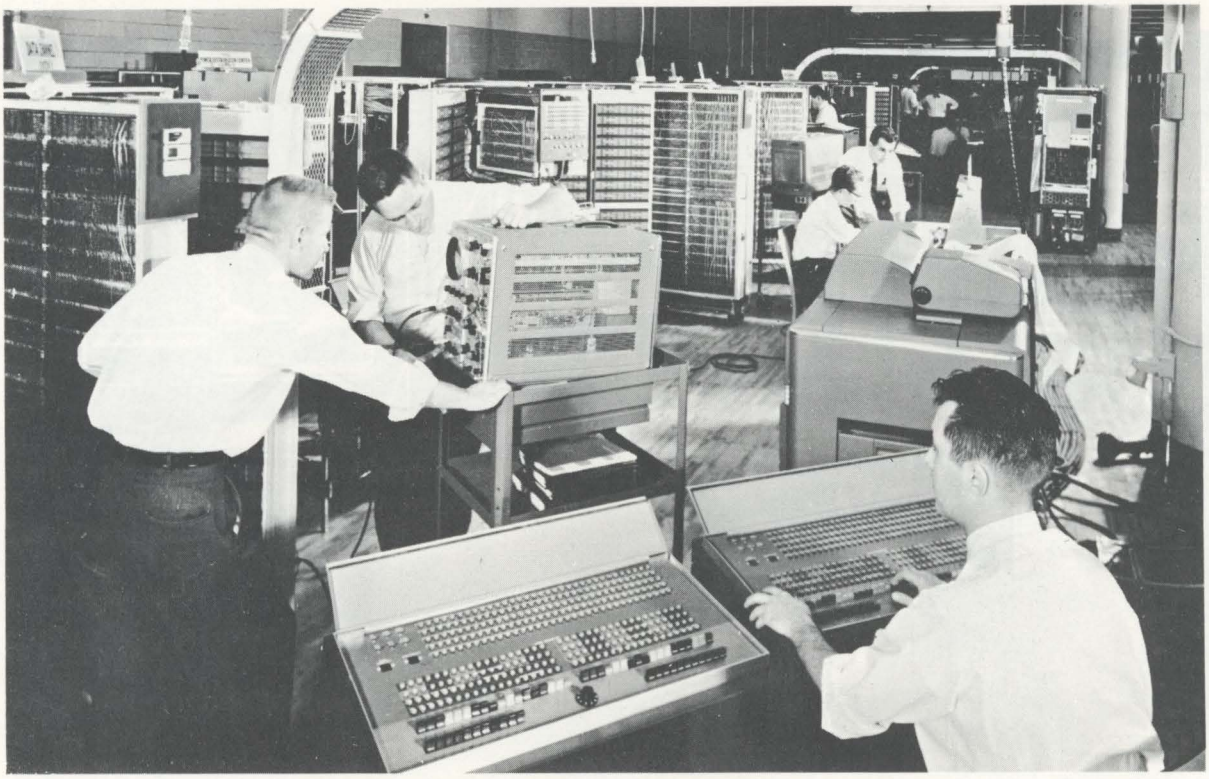


Figure 2. In-manufacture Test of Large-scale IBM Data Processing System

than 50 percent in the past 10 years, while employment in all manufacturing has increased by only six percent.

Despite these fundamental changes in our economy, clerical mechanization has not kept pace with production line developments in the factory.

Great opportunities and challenges lie ahead. An example of what can be done is the development of magnetic character sensing for the banking industry. The estimated 10 billion checks that circulate annually in the United States present a staggering task in data handling for banks. Each check drawn on a bank must be handled at least six times before it is cancelled and returned. Even when business machines were introduced to handle part of this chore, operators were needed to transfer data from the checks to a form in which the data could be used by the machines.

Magnetic character sensing, developed by computer manufacturers in cooperation with the American Bankers Association (ABA), permits data to be read directly by both man and machine (Figure 3). By agreement among computer manufacturers, check printers, and the ABA, banking documents such as checks, deposit slips, and debit and credit memos can be printed in magnetic ink. Printed information about the bank of origin, depositor's account number and other essential data can be read directly by the machine. Only the specific amount of each check or deposit slip need be recorded on the document in magnetic print. And this need be done only once by an operator to process the document through its whole routine.

In addition to the growing need for mechanization of clerical routines and management procedures, there is the tremendously expanded need for data processing to match the new rate of technological growth and scientific research. The demands for information are enormous. More and more, data processing systems

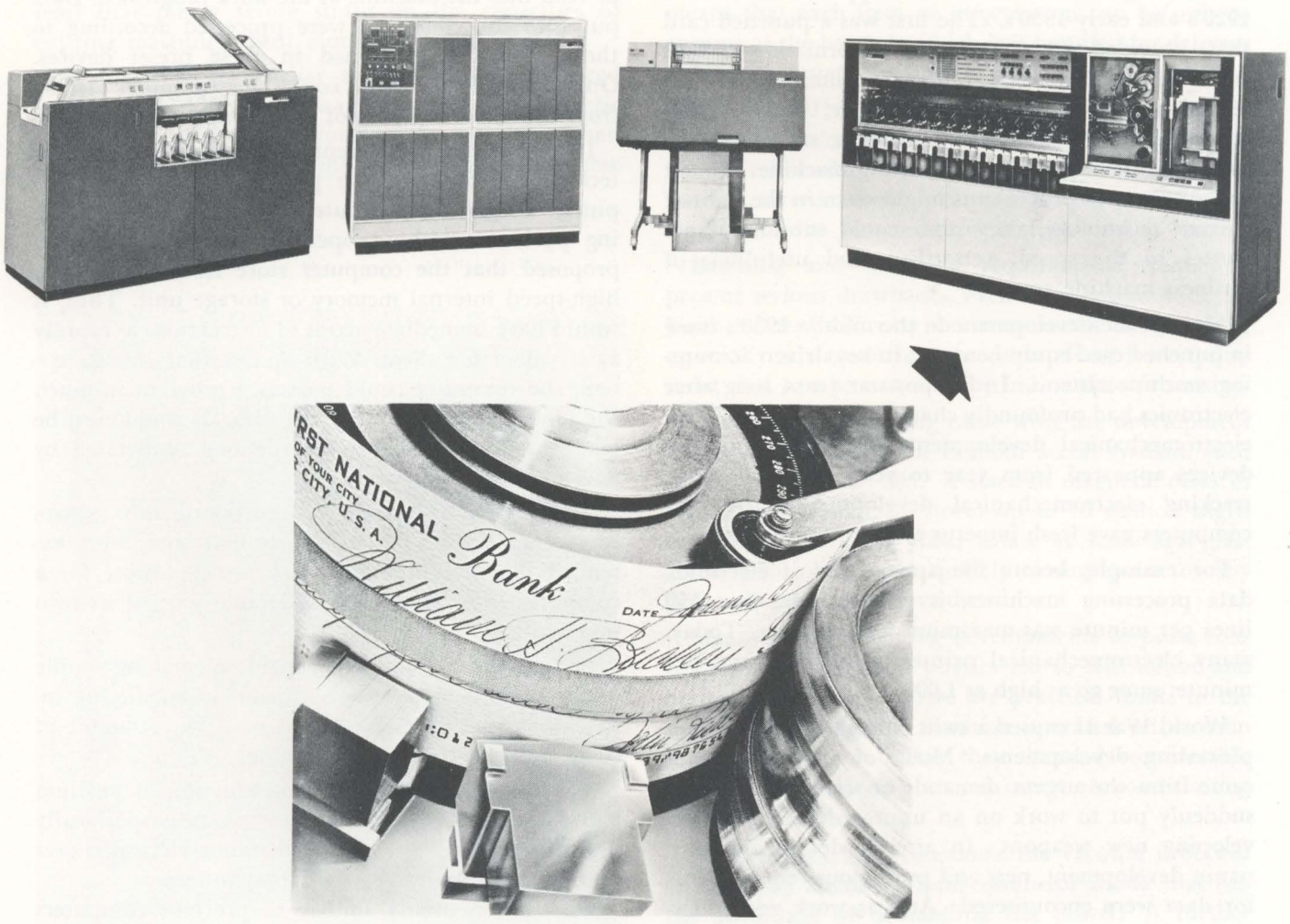


Figure 3. Magnetic Character Sensing — IBM 1210 Reader Sorter with IBM 1401 Data Processing System

are depended on for information to run enterprises, administer institutions, direct research, and plan endeavors.

Regardless of the product or problem, the nature of the enterprise or institution, wherever there is need for information upon which human judgments can be based, there may also exist a need for a data processing machine.

The Growth of IBM Data Processing

Although data processing equipment is a tool of astonishing versatility, the automatic processing of data is so recent that only 30 years are needed to trace its biggest period of growth.

Punched cards were introduced during the census of 1890, but the data processing industry, as recently as 1930, amounted to little more than a fledgling, although a lively one.

Three significant achievements appeared in the late 1920's and early 1930's. The first was a punched card that provided 80 columns of information—almost twice the capacity of the older 45-column cards. The second was the automatic multiplier; previous machines had been able only to add or subtract. The third was the alphabetic accounting machine. Modest though these improvements might seem in the light of current technology, they represented substantial advances in the speed, versatility, and usefulness of business machine systems.

Most of the development in the middle 1930's came in punched card equipment and in key driven accounting machine systems. In the postwar years, long after electronics had profoundly changed the industry, these electromechanical developments continued and new devices appeared from year to year. Far from sidetracking electromechanical developments, the new computers gave fresh impetus to advances in this field.

For example, before the appearance of electronic data processing machines, it was thought that 150 lines per minute was maximum for a printer. Today, many electromechanical printers print 500 lines per minute; some go as high as 1,000.

World War II caused a swift change of pace in data processing developments. Much of the momentum came from the urgent demands of science which was suddenly put to work on an unprecedented scale developing new weapons. In aircraft design and ordnance development, new and prodigious requirements for data were encountered. And as work got underway on the atomic bomb, scientists found themselves faced with new dimensions in calculation.

Both here and abroad, the first two large scale computers were developed in university laboratories. The earliest, the ENIAC, came from the University of Pennsylvania; Europe's first, the EDSAC, came from the laboratories of Cambridge University in England.

In these machines, the switching and control functions, once entrusted to relays, were handled by vacuum tubes. Thus, the relatively slow movements of switches in electromechanical computers were replaced by the swift motion of electrons. By this changeover, it became possible to increase the speed of calculation and perform computations 1,000 times as fast as before.

Almost concurrently with the use of electronics came another major development that was to widen the capabilities of data processing systems and expand their opportunities for application. This advance is embodied in what is called a stored program computer. At the start, machine instructions were programmed on interchangeable control panels, cards, or paper tapes. Detailed instructions had to be wired in or read into the machine as the work progressed. Data put into the computer were processed according to the instructions contained in these preset devices. Only in a limited way could the computer depart from the fixed sequence of its program.

It soon became apparent that these programming techniques inhibited the performance of the computer. To give the computer greater latitude in working problems without operator assistance, scientists proposed that the computer store its program in a high-speed internal memory or storage unit. Thus, it would have immediate access to instructions as rapidly as it called for them. With an internal storage system, the computer could process a program in much the same way that it processed data. It could even be made to modify its own instructions as dictated by developing stages of work.

The earliest computer to incorporate this feature was completed in 1948. Subsequent computers extended the principle until it became possible for a computer to generate a considerable part of its own instructions.

Because the computer is capable of making simple decisions, and because it is capable of modifying instructions, the user is relieved of a vast amount of costly and repetitive programming.

The early 1950's saw the introduction of medium and large scale data processing systems, specifically designed to take over the burdensome clerical chores that beset so many growing companies.

Though essentially similar to previous computers in the way they processed data, these new business systems differed substantially in various parts of their

make-up. In scientific research, most problems call for relatively small numbers of items to be subjected to intensive machine processing. In business operations, the reverse is more often true. Here the need is for machines that accommodate vast numbers of items while the processing, by comparison, is ordinarily quite simple.

Modifications in these new business systems were addressed to the twin problems of input and output.

Early computers had used punched cards and paper tapes for the input of information. Now a method was perfected for storing information as magnetized spots on magnetic tape. This new technique provided input speed 50 to 75 times that of cards. It brought improvement in input, output, and storage.

After the Korean War, man's need always seemed to be one jump ahead of the computer's ability to handle the logical and arithmetic labors of his reasoning. The demand quickened especially in such fields as nuclear physics and space technology where work on the H-bomb and ballistic missiles presented problems that put a severe strain on the capacities of existing machines. Still more speed was needed.

A substitute for earlier storage devices appeared in the early 1950's—the magnetic core. A magnetic core is a small ring of ferromagnetic material. When strung on a complex of fine wires (Figure 4), these cores can be made up into a high-speed internal storage system. An array of cores—some magnetized in one direction, some in the other—represents items of information. Items in a core storage can be located and made ready for processing in a few millionths of a second.

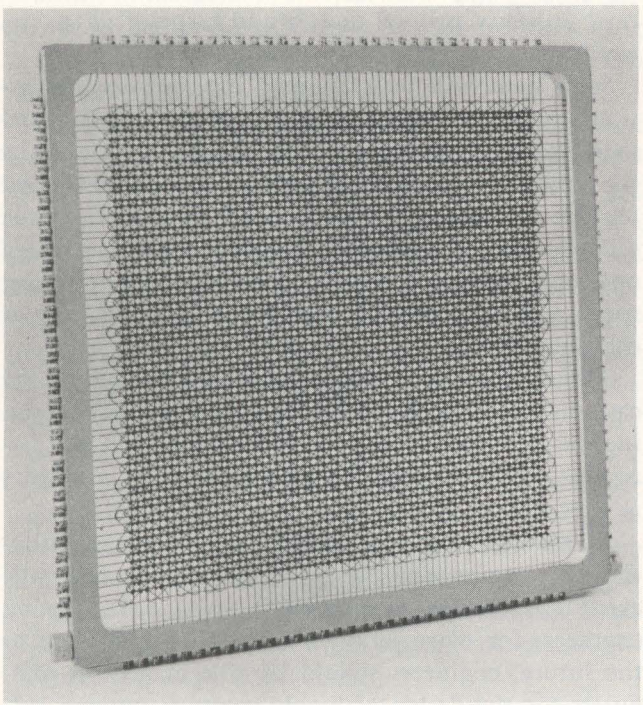


Figure 4. Magnetic Core Plane

Almost at the same time, other engineers perfected magnetic drum storage. Access time on the drum was substantially slower than that on the core system, but storage capacity was substantially increased. And while access time on the drum was slower than that of the core, it was faster than that of magnetic tape.

Other conditions peculiar to business led to still more developments. A major one is a system that overcomes a problem—batching—often encountered in data processing. For example, when magnetic tapes are used to store information in a computer, the user must accumulate information in batches before putting it on tape. Otherwise, the computer would be prohibitively costly and time consuming. But when this limitation is applied to business practice, it means that each item of information can be only as current as the batch in which it is bundled for delivery to the computer. In ordinary operations, hours and sometimes days may elapse between batches.

The limitation is compounded when the user calls for the retrieval of a piece of information. The computer is forced to search through a long reel of information for the piece. Access is slow; time is lost.

Batching and searching requirements frequently present serious drawbacks, even in scientific work. In business, the difficulty becomes much more acute, especially in accounting procedures. Here is a requirement that can be met only by in-line data processing.

In-line data processing came with the development in the middle 1950's of random access systems, such as the IBM RAMAC® 305. A stack of magnetic disks in the RAMAC (Figure 5) stores up to 10 million digits of information. The disks rotate at 1,200 RPM past access arms that move quickly to any point of any disk to deposit or retrieve data.

Meanwhile, continuing developments in pulse electronics and solid state physics led to still newer and better components. There are practical limits to the size and capacity of a machine operated on vacuum tubes. Tubes are bulky; they demand considerable power; they produce heat and create air conditioning problems.

In some switching functions, the vacuum tube was replaced by a smaller semi-conductor diode that has the advantage of demanding less power. A further advance came when tiny transistors were introduced in place of the vacuum tubes in the computer. Not

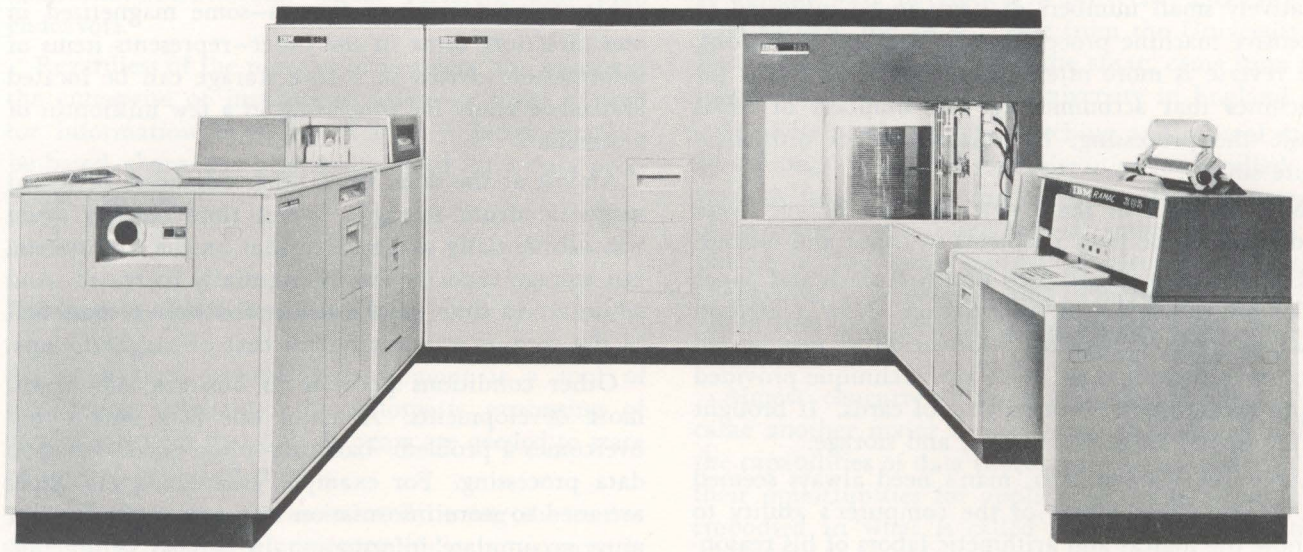


Figure 5. IBM RAMAC® 305

only can these transistors be packed into smaller units (Figure 6), but they have greater reliability. The change-over to transistors is now accomplished.

Research scientists have already advanced to still further stages in design. Some are studying the use of microwave phenomena as a medium for performing computer logic. Others are studying the behavior of materials and electrons at extremely low temperatures (cryogenics).

As always, the objective is to develop a better, more versatile, more useful computer—one that will work faster, store more information, demand fewer instructions, require less power, and occupy less space.

The Future

Future computers will inevitably introduce changes in the way we work, in the way we learn, and even in the way we provide for our armed defense. Research even now points the way to some changes which may be nearer at hand than we suspect.

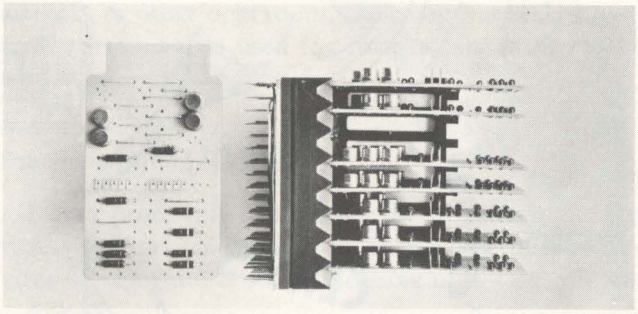
For the moment, computer users are handicapped in communicating with these machines. Instructions must be coded laboriously into machine language. Instructions conveyed in a few spoken words to a human being may require hundreds of logical movements in a computer—and this is a problem because the computer must be instructed in each movement.

The new science of automatic programming seeks to make programming easier and more manageable. The goal is to build and program computers so that they accept instructions in everyday English. Ultimately, computer scientists hope to develop machines that read ordinary printed matter and respond to spoken words.

Soon there will be a common national—or international—machine language for computers. With a common "tongue," machines will be designed to operate by the same instructional language. A program generated for one machine can then be used by any other; the difficult task of creating a program need be tackled only once. Greater cooperation will be possible among users, and progress will be speeded in the science of computer use.

Already there is one significant step forward in the case of machines that read. Magnetic character sensing, as previously indicated, utilizes numbers and letters printed in a way that can be read by machines as well as man.

Another development points to further mechanization in engineering design work. Computers will eventually assume routine engineering tasks, freeing engineers for more productive work. Not too far in the future, engineers should be able to call on computers for the design of highly complex systems. All that will be required is a statement of the engineering



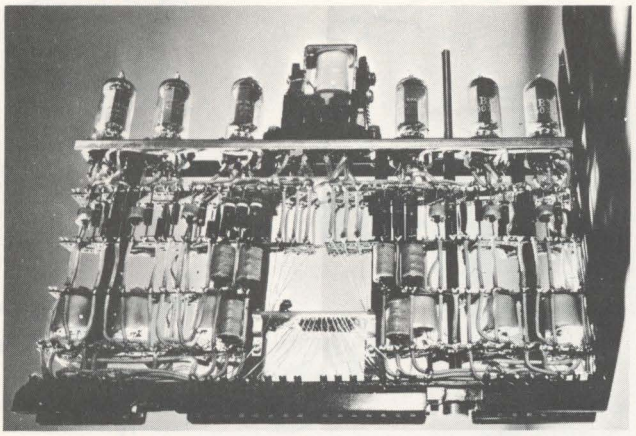


Figure 6. Computer Circuitry, Solid State and Vacuum Tube

problem in mathematical equations. The greatest difficulty in achieving this technique is that of stating design criteria unequivocally and in a form that can be programmed into a computer.

Computers, by 1980, will probably be quite different from today's. Storage and processing units as powerful as today's largest may be the size of a television set, perhaps smaller. Already there are systems where a single computer serves a number of inquiry stations. Such systems can be expanded to produce larger networks and integrate widely scattered business operations. Beyond that, it may be possible to query the computer merely by talking into a microphone. Eventually, the computers may even be able to answer by voice.

Among the problems that challenge systems engineers is the paperwork that gluts so many administrative channels. Information, they say, can be communicated and processed more accurately and cheaply by network-integrated data processing systems. Some even contemplate a time when paper bank checks will disappear in many transactions.

Instead of handing an employee his paycheck, a paymaster in 1985 may simply instruct a computer, housed in a bank, to credit the employee's earnings to his bank account.

At this point, a chain reaction begins. The company's account is debited by the amount of the paycheck. This information goes to the company's accounting machines for processing and storage. Now, assume that the employee wants to buy something that costs more than he wishes to pay for in cash. The salesclerk asks for identification, a card that identifies the customer to the salesclerk and to the computer. The clerk instructs the bank computer to debit the customer's account by the amount of his purchases. The same amount is credited automatically to the store's account. Information on the sale goes to the store's computer for processing with sales, financial, and inventory data. Periodically, the customer gets a statement on his deposits and computer transactions.

Still other scientists have turned their talents to developing what they prophetically call information centers.

The science of information retrieval by machine is under rigorous study. As our society grows, the information it generates will increase; the task of finding what one is looking for will become increasingly difficult and time consuming.

Information centers of the future would collect, catalog, and retrieve information electronically by machine. Queried about a subject, a center could provide specific source references, cross-references, and subdivisions of subject matter. Or, asked a question, it could give a direct answer.

Such a development would mean immediate access to more information than was ever before instantly available. A man might never have the ability to know all that he might want to know, but he would have the means of finding it as he had need.

Everywhere the pace of technological advance has quickened. During the next 25 years, man will journey into space; he may reach the nearest planets. He will look about him on earth. He will pose questions and search for knowledge. He will make discoveries not yet imagined.

In such a world and at such a time, man will have greater and more pressing need for business machines than ever before. These needs, in turn, will dictate machine advances.

What will these advances mean? They signify continuing and hopeful advances in human progress. And because these machines have so many applications, because they can be used in so many ways, they hold out the promise of progress in the use we make of our time, progress in the way we employ our talents, progress in the search we make for learning and knowledge. Put all these things together, and it may not

be too much to hope that these machines, in some way, will help us to keep those very values that make the human being so distinct and his life so worthwhile.

The Data Processing System

Data processing is a series of planned actions and operations upon information to achieve a desired result. The procedures and devices used constitute a data processing system (Figure 7). The devices may vary: all operations may be done by machine, or the devices may be only pencil and paper. The procedures, however, remain basically the same.

There are many types of IBM data processing systems. These systems vary in size, complexity, speed, cost, and application. But, regardless of the information to be processed or the equipment used, all data processing involves at least three basic considerations:

1. The source data or *input* entering the system.
2. The orderly, planned *processing* within the system.
3. The end result or *output* from the system.

Input may consist of any type of data: commercial, scientific, statistical, engineering, and so on (Figure 8).

Processing is carried out in a pre-established sequence of instructions that are followed automatically by the computer (Figure 9). The plan of processing is always of human origin. By calculation, sorting,

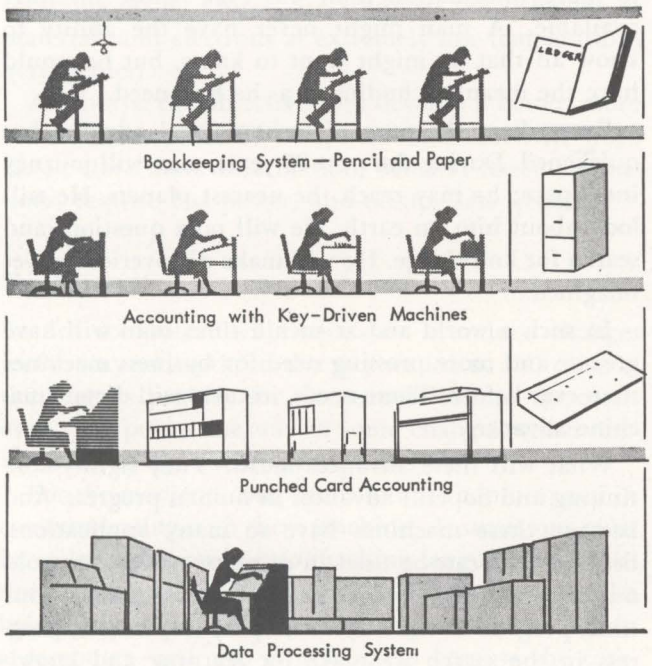


Figure 7. Data Processing Systems

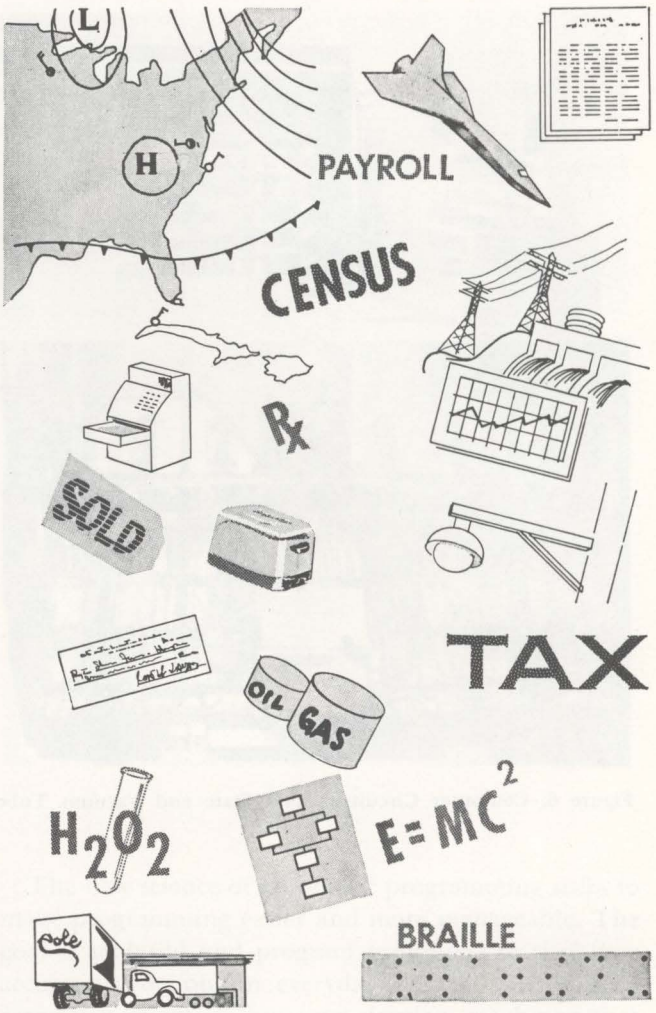


Figure 8. Sources of Data

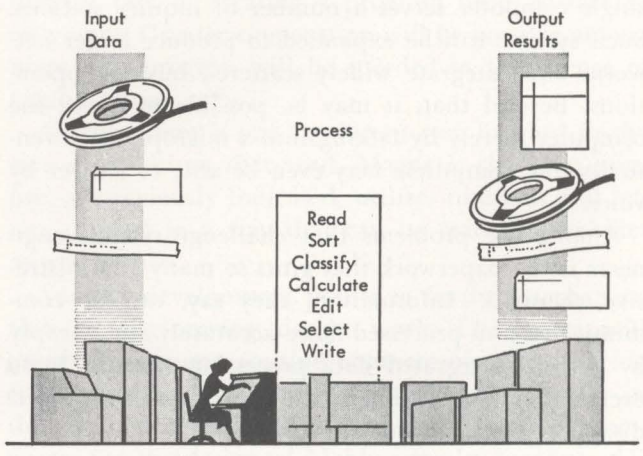


Figure 9. Data Processing by Computer

analysis, or other operations, the computer arrives at a result that may be used for further processing or recorded as reports or files of data.

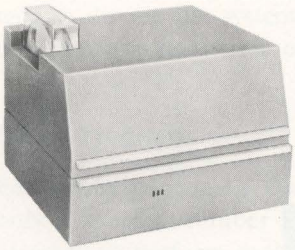
Functional Units

All data processing systems can be divided into four types of functional units: input devices, output devices, storage, and central processing unit.

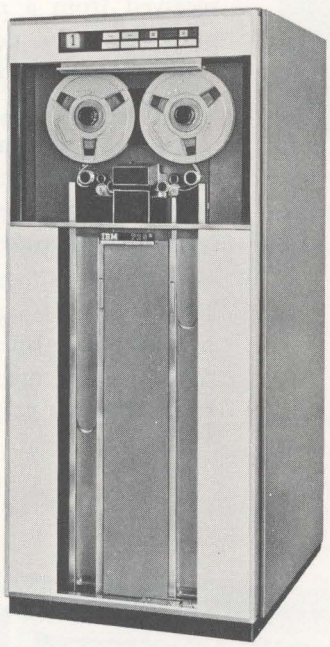
Input and Output Devices

The data processing system requires, as a necessary part of its information-handling ability, features that can enter data into the system and record data from the system. These functions are performed by input-output devices (Figure 10) linked directly to the system.

Input devices read or sense coded data that are recorded on a prescribed medium and make this in-



IBM 962 Tape Punch



IBM 729 IV Magnetic Tape Unit

IBM 1403 Printer

IBM 382 Paper Tape Reader

IBM 7500 Card Reader

IBM 7550 Card Punch

Figure 10. Input-Output Devices

formation available to the computer. Data for input are recorded in IBM cards and paper tape as punched holes; on magnetic tape, as magnetized spots along the length of the tape; and on paper documents as characters printed in magnetic ink.

The method of recording data for machine use and the characteristics of each medium are discussed in later chapters.

Output devices record or write information from the computer on IBM cards, paper tape, magnetic tape, or as printed information on paper. The number and type of input-output devices connected directly to the computer depend on the design of the system and its application.

Special data conversion operations are associated with all computer systems to transcribe information recorded on one medium to another. For example, information on punched cards can be transcribed automatically to magnetic tape. This operation may take place *on-line,* utilizing the computer, or *off-line,* utilizing input-output devices independently.

Storage

Storage is somewhat like an electronic filing cabinet, completely indexed and almost instantaneously accessible to the computer (Figure 11).

All data must be placed in storage before they can be processed by the computer. Information is read into storage by an input unit and is then available for internal processing. Each location, position, or section of storage is numbered so that the stored data can be readily located by the computer as needed.

The computer may rearrange data in storage by sorting or combining different types of information received from a number of input units. The computer may also take the original data from storage, calculate new information, and place the result back in storage.

The size or capacity of storage determines the amount of information that can be held within the system at any one time. In some computers, storage capacity is measured in millions of digits or characters,

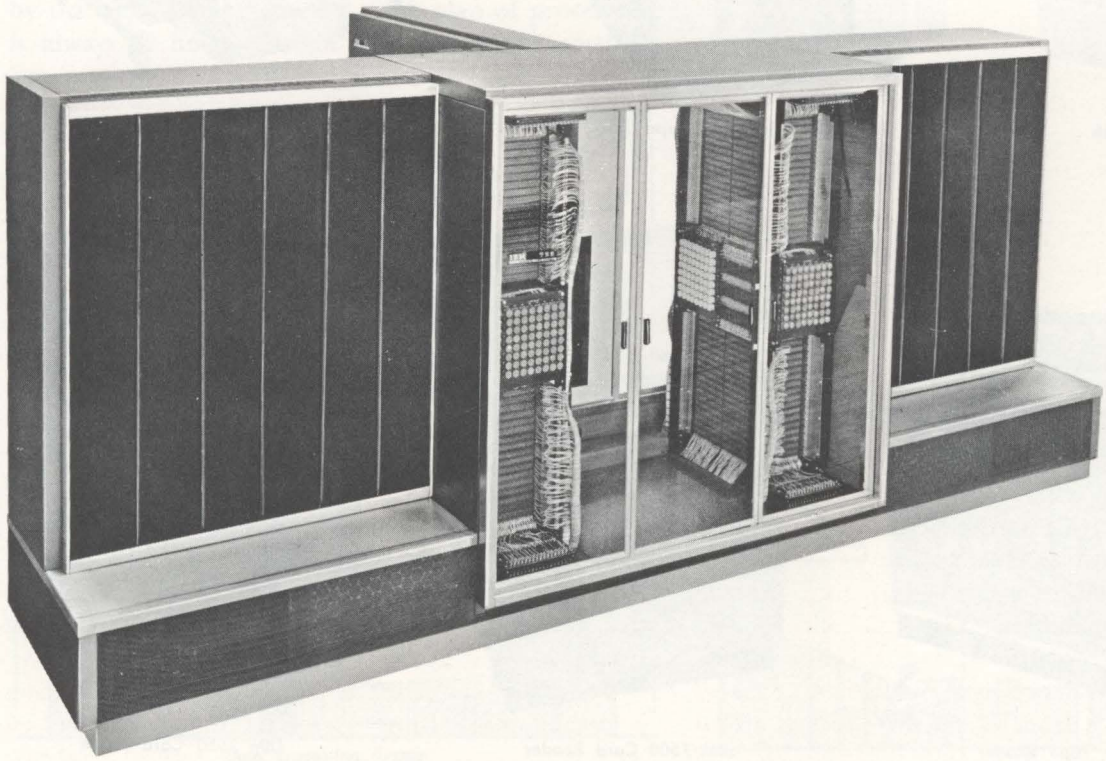


Figure 11. IBM 738 Magnetic Core Storage

providing space to retain entire files of information. In other systems, storage is smaller and data are held only while being processed. Consequently, the capacity and design of storage affect the method in which data are handled by the system.

Central Processing Unit

The central processing unit (Figure 12) is the controlling center of the entire data processing system. It can be divided into two parts:

1. The arithmetic-logical unit.
2. The control section.

The arithmetic-logical unit performs such operations as addition, subtraction, multiplication, division, shifting, transferring, comparing, and storing. It also has logical ability—the ability to test various conditions encountered during processing and to take action called for by the result.

The control section directs and coordinates the entire computer system as a single multipurpose machine. These functions involve controlling the input-output units and the arithmetic-logical operation of the central processing unit, and transferring data to and from storage, within given design limits. This section directs the system according to the procedure originated by its human operators.

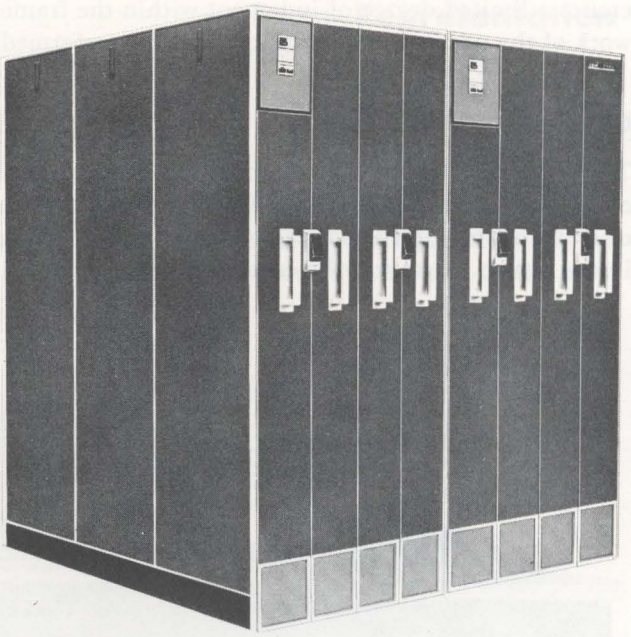


Figure 12. IBM 7100 Central Processing Unit

Stored Programs

Each data processing system is designed to perform only a specific number and type of operations. It is directed to perform each operation by an instruction. The instruction defines a basic operation to be performed and identifies the data, device, or mechanism needed to carry out the operation. The entire series of instructions required to complete a given procedure is known as a program.

For example, the computer may have the operation of multiplication built into its circuits in much the same way that the ability to add is built into a simple desk adding machine. There must be some means of directing the computer to perform multiplication just as the adding machine is directed by depressing keys. There must also be a way to instruct the computer where in storage it can find the factors to multiply.

Further, the comparatively simple operation of multiplication implies other activity that must precede and follow the calculation. The multiplicand and multiplier must be read into storage by an input device. This device must previously have had access to the record or records from which these factors are to be supplied. Once the calculation is performed, the product must be returned to storage at a specified location, from which it may be written out by an output device.

Any calculation, therefore, involves reading, locating factors in storage, perhaps adjusting the result, returning the result to storage, and writing out the completed result. Even the simplest portion of a procedure involves a number of planned steps that must be spelled out to the computer if the procedure is to be accomplished.

An entire procedure is composed of these individual steps grouped in a sequence that directs the computer to produce a desired result. Thus, a complex problem must first be reduced to a series of basic machine operations before it can be solved. Each of these operations is coded as an instruction in a form that can be interpreted by the computer and is placed in the main storage unit as a *stored program*.

The possible variations of a stored program provides the data processing system with almost unlimited flexibility. One computer can be applied to a great number of different procedures by simply reading in or *loading* the proper program into storage. Any of the standard input devices can be used for this purpose, because instructions can be coded into machine language just as data can.

The stored program is accessible to the machine, providing the computer with the ability to alter its own program in response to conditions encountered during an operation. Consequently, the machine ex-

ercises a limited degree of judgment within the framework of the possible operations that can be performed.

Console

The console (Figure 13) provides external control of the data processing system. Keys turn power on or off, start or stop operation, and control various devices in the system. Data may be entered directly by manually depressing keys. Lights are provided so that data in the system may be visually displayed. The system may also be operated from the console to trace or check out a procedure one step at a time.

On some systems, a console typewriter provides limited output. The typewriter may print messages, signaling the end of processing or an error condition. It may also print totals or other information that enable the operator to monitor and supervise operation.

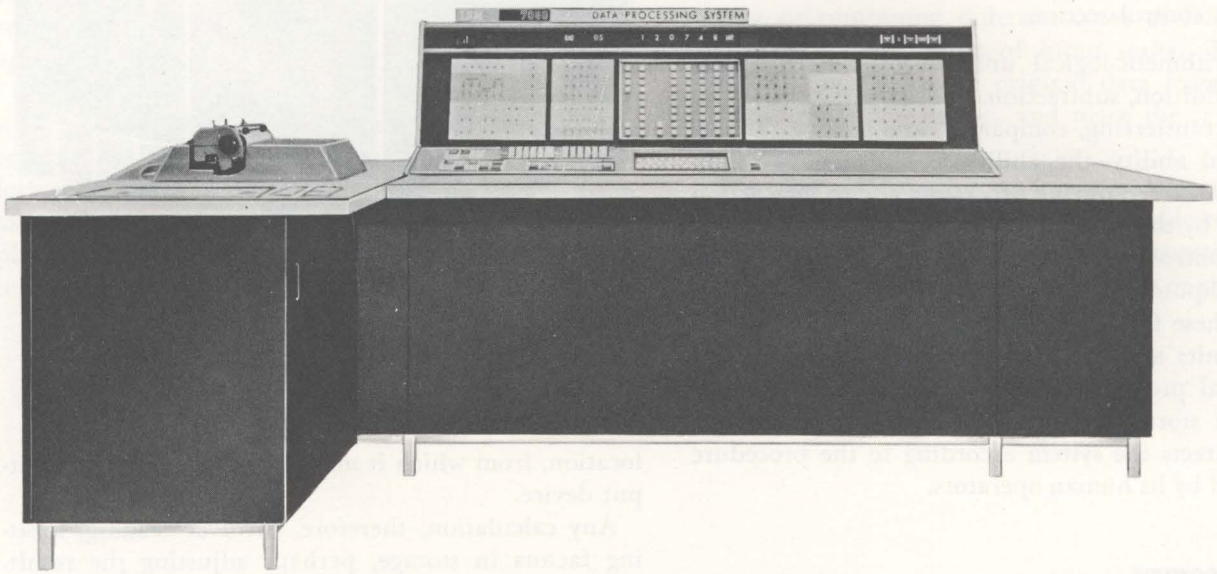


Figure 13. IBM 7153 Console

Symbols convey information; the symbol itself is not the information but merely represents it. The printed characters on this page are symbols and, when understood, convey the writer's meaning.

The meaning of symbols is one of convention. A symbol may convey one meaning to some persons, a different meaning to others, and no meaning to those who do not know its significance (Figure 14).

Presenting data to the computer system is similar in many ways to communicating with another person by letter. The intelligence to be conveyed must be reduced to a set of symbols. In the English language, these are the familiar letters of the alphabet, numbers, and punctuation. The symbols are recorded on paper in a prescribed sequence and transported to another person who reads and interprets them.

Similarly, communication with the computer system requires that data be reduced to a set of symbols that can be read and interpreted by data processing machines. The symbols differ from those commonly used by people, because the information to be represented must conform to the design and operation of the machine. The choice of these symbols and their meaning is a matter of convention on the part of the designers. The important fact is that information can be represented by symbols, which become a language for the communication between people and machines.

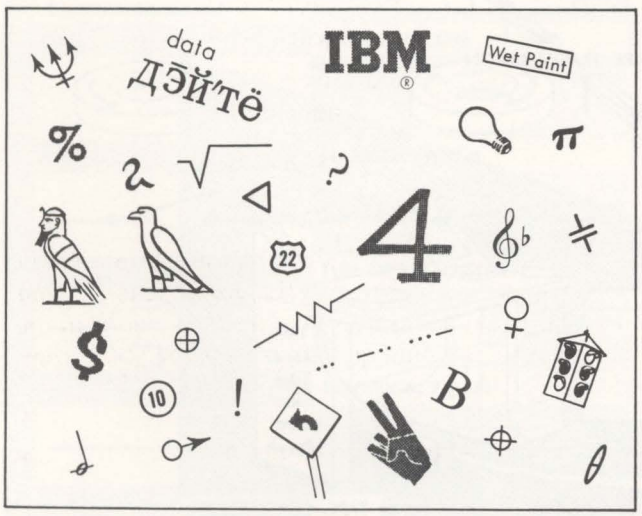
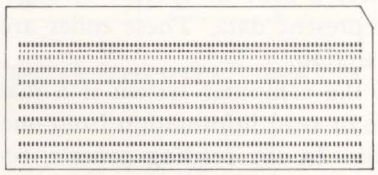
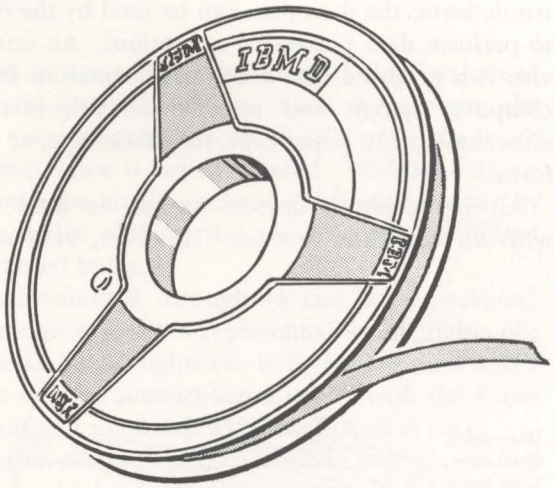


Figure 14. Symbols for Communication

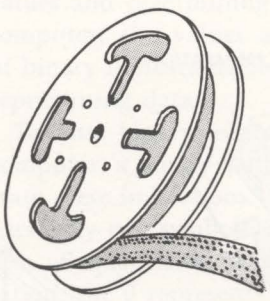
Information to be used with the computer systems can be recorded on four media: IBM cards, paper tape, magnetic tape, and magnetic ink characters (Figure 15). Data are represented on the IBM card by the pres-



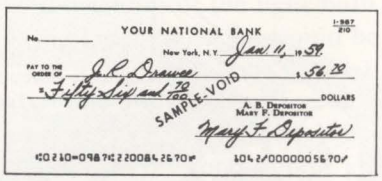
IBM Card



Magnetic Tape



Paper Tape



Magnetic Ink Characters

Figure 15. Data Recording Media

ence or absence of small rectangular holes in specific locations of the card. In a similar manner, small circular holes along a paper tape represent data. On magnetic tape, the symbols are small magnetized areas, called spots or bits, arranged in specific patterns. Magnetic ink characters – the arabic numerals 0 to 9 and four special characters – are printed on paper. The shape of the characters and the magnetic properties of the ink permit the printed data to be read by both man and machine.

Each medium requires a code or specific arrangement of symbols to represent data. These codes are described later in this section.

An input device of the computer system is a machine designed to sense or read information from one of the recording media. In the reading process, recorded data are converted to or symbolized in electronic form; the data then can be used by the machine to perform data processing operations. An output device is a machine that receives information from the computer system and records the information on either IBM cards, paper tape, magnetic tape, or printed forms.

All input-output devices cannot be used directly with all computer systems. However, data recorded on one medium can be transcribed to another medium for use with a different system. For example, data on IBM cards or paper tape can be transcribed onto magnetic tape. Conversely, data on magnetic tape can be converted to cards, paper tape, or printed reports.

As there is communication between people and machines, there is also communication from one machine to another (Figure 16). This intercommunication may be the direct exchange of data, in electronic form, over wires, cables, or radio waves; or, recorded output of one machine or system may be used as input to another machine or system.

Computer Data Representation

Not only must there be a method of representing data on IBM cards, paper tape, magnetic tape, and in magnetic ink characters, but there must also be a method of representing data within a machine.

In the computer, data are represented by many electronic components: vacuum tubes, transistors, magnetic cores, wires, and so on. The storage and flow of data through these devices are represented as elec-

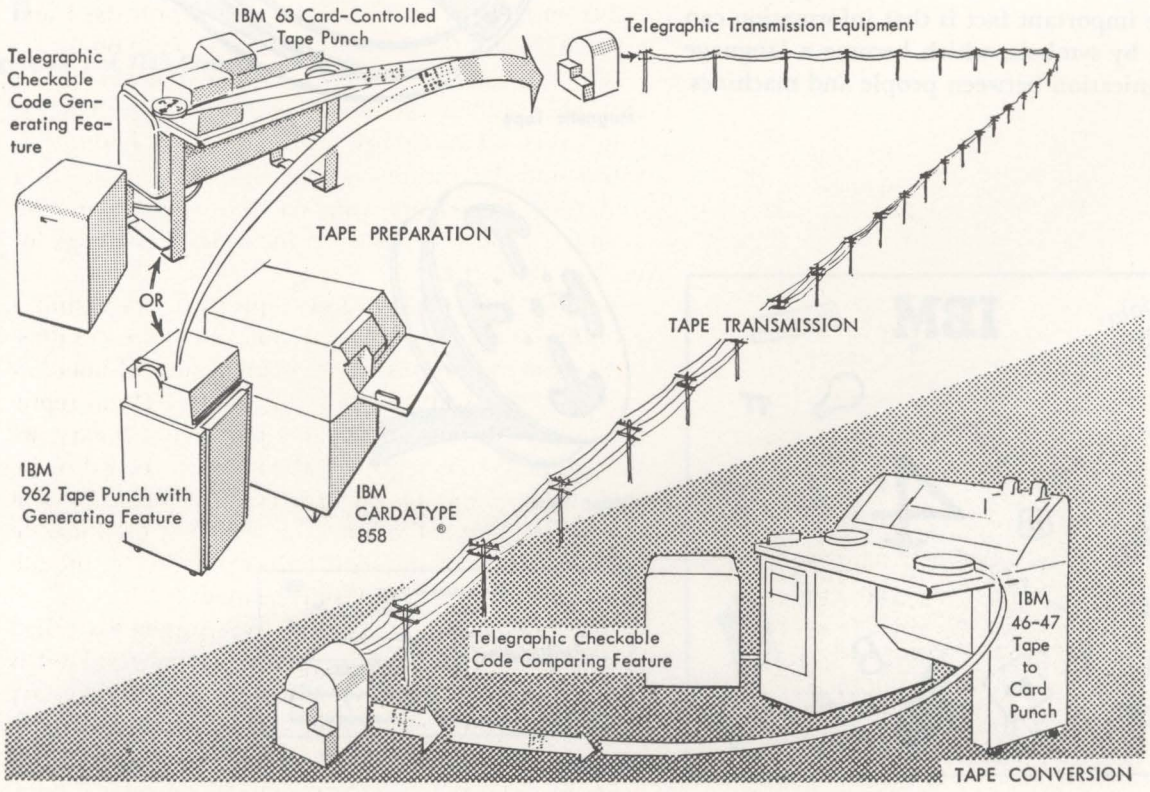


Figure 16. Machine-to-Machine Communication

tronic signals or indications. The presence or absence of these signals in specific circuitry is the method of representing data, much as the presence or absence of holes in an IBM card represents data.

Binary Mode

Computers function in what is called a binary mode. This term simply means that the computer components can indicate only two possible states or conditions. For example, the ordinary light bulb operates in a binary mode: it is either on, producing light; or it is off, not producing light. The presence or absence of light indicates whether the bulb is on or off. Likewise, within the computer, vacuum tubes or transistors are maintained either conducting or nonconducting; magnetic materials are magnetized in one direction or in an opposite direction; and specific voltage potentials are present or absent (Figure 17). The binary modes of operation of the components are signals to the computer, as the presence or absence of light from an electric light bulb is to a person.

Representing data within the computer is accomplished by assigning or associating a specific value to a binary indication or group of binary indications. For example, a device to represent decimal values could be designed with four electric light bulbs and switches to turn each bulb on or off (Figure 18).

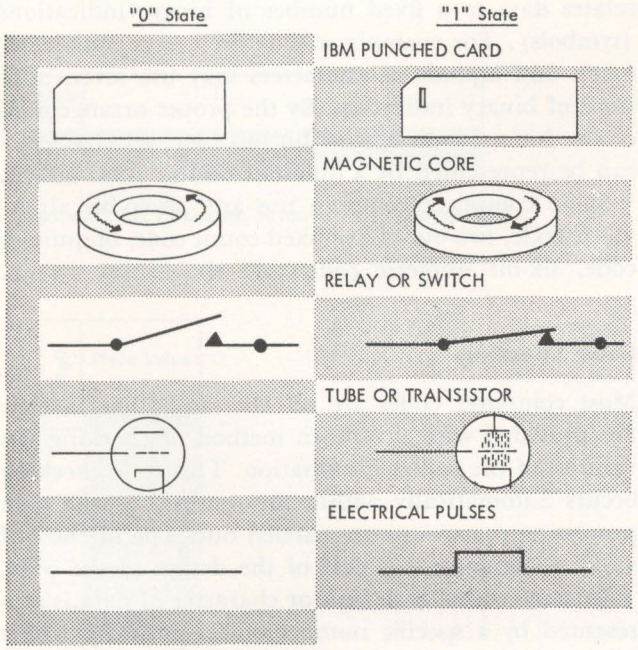


Figure 17. Binary Indicators

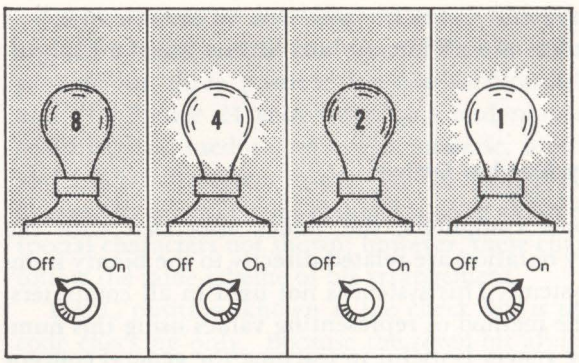


Figure 18. Representing Decimal Data

The bulbs are assigned arbitrary decimal values of 1, 2, 4, and 8. When a light is on, it represents the decimal value associated with it. When a light is off, the decimal value is not considered. With such an arrangement, the single decimal value represented by the four bulbs will be the numeric sum indicated by the lighted bulbs.

Decimal values 0 through 15 can be represented. The numeric value 0 is represented by all lights off; the value 15, by all lights on; 9, by having the 8 and 1 lights on and the 4 and 2 lights off; 5, with the 1 and 4 lights on and the 8 and 2 lights off; and so on.

The value assigned to each bulb or indicator in the example could have been something other than the values used. This change would involve assigning new values and determining a scheme of operation. In a computer, the values assigned to a specific number of binary indications become the code or language for representing data.

Because binary indications represent data within a computer, a binary method of notation is used to illustrate these indications. The binary system of notation uses only two symbols, zero (0) or one (1), to represent all quantities. In any one position of binary notation, the 0 represents the absence of a related or assigned value and the 1 represents the presence of a related or assigned value. For example, to illustrate the indications of the light bulb in Figure 18, the following binary notation would be used: 0101.

The binary notations 0 and 1 are commonly called bits. The 0 bit is described as no bit and the 1 bit is described as a bit. Although 0 or 1 bits are necessary to illustrate the condition of a binary indication or a group of binary indications, the 1 bits are the bits generally referred to. For example, the binary notation 0101 of Figure 18 would be described as having a

bit in the 1 and 4 bit positions. The assumption is that there are no bits (0 bits) in the 2 and 8 bit positions.

Binary Number System

In some computers, the values associated with the binary notation are related directly to the binary number system. This system is not used in all computers, but the method of representing values using this numbering system is useful in learning the general concept of data representation.

The common decimal number system uses ten symbols or digits to represent all quantities, and the place value of the digits signifies units, tens, hundreds, thousands, and so on. The binary or base-two number system uses only two symbols or digits: 0 and 1. The position value of the bit symbols (0 or 1) is based on the progression of powers of 2; the units position of a binary number has the value of 1; the next position, a value of 2; the next, 4; the next, 8; the next, 16; and so on (Figure 19).

8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Figure 19. Place Value of Binary Numbers

In pure binary notation, the binary digits or bits indicate whether the corresponding power of 2 is absent or present in each position of the number. The 1 bit represents the presence of the value and the 0 bit represents the absence of the value. The place value of the digits does not signify units, tens, hundreds, or thousands, as in the decimal system; instead, the place value signifies units, twos, fours, eights, sixteens, and so on. Using this system the quantity 12, for example, is expressed with the symbols 1100, meaning $(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0)$, or $(1 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$.

Figure 20 shows the binary representation of the decimal values 0 through 16. Note that the decimal digits 0 through 9 are expressed by four binary digits. The system of coding or expressing decimal digits in an equivalent binary value is know as binary coded decimal (BCD). For example, the decimal value 265,498 would appear in binary coded decimal form as shown in Figure 21.

Decimal Value	Place Value 16	8	4	2	1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0

Figure 20. Binary Representation of Decimal Values 0-16

Decimal Digits	2			6			5			4			9			8		
Binary Value	0	0	1	0	0	1	1	0	1	0	1	0	1	0	0	1	0	0
Place Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

Figure 21. Binary Coded Decimal

Computer Codes

The method or system used to represent (symbolize) data is known as a code. In the computer, the code relates data to a fixed number of binary indications (symbols). For example, a code used to represent numeric and alphabetic characters may use seven positions of binary indication. By the proper arrangement of the binary indications (bit, no bit), all characters can be represented by a different combination of bits.

Some computer codes in use are: seven-bit alphameric code, two-out-of-five fixed count code, bi-quinary code, six-bit numeric code, and the binary system.

Code Checking

Most computer codes are self-checking; that is, they are provided with a built-in method of checking the validity of the coded information. This code checking occurs automatically within the machine as the data processing operations are carried out. The method of validity checking is a part of the design of the code.

In some codes, each unit or character of data is represented by a specific number of bit positions which must always contain an even number of 1 bits. Different characters are made up of different combinations

of 1 bits, but the number of 1 bits in any valid character is always even. With this code system, a character with an odd number of 1 bits is detected and an error is indicated. Likewise, a code may be used in which all characters must have an odd number of 1 bits; an error is indicated when characters with an even number of 1 bits are detected.

This type of checking is known as a parity check. Codes which use an even number of 1 bits are said to have even parity. Codes which use an odd number of bits are said to have odd parity.

In other codes, the number of 1 bits present in each unit of data is fixed. For example, a code may use five bit positions to code all digits but only two 1 bits will be present in each digit. Digits having more or fewer than two 1 bits cause an error indication. This system of checking is known as a fixed count check.

Seven-Bit Alphameric Code (Binary Coded Decimal)

In this code, all characters – numeric, alphabetic, and special–are represented (coded) using seven positions of binary notation. These positions are divided into three groups: one check position, two zone positions, and four numeric positions (Figure 22).

The four numeric positions are assigned decimal values of 8, 4, 2, and 1 and represent, in binary coded decimal form, the numeric digits 0 through 9 (Figure 23). Note that 0 is represented as 1010, actually the binary number for 10. The B and A zone bits are not present (00) when the numeric digits 0 through 9 are represented.

Check Bit	Zone Bits		Numeric Bits			
C	B	A	8	4	2	1

Figure 22. Bit Positions, Seven-Bit Alphameric Code

Decimal Digit	Place Value 8	4	2	1
0	1	0	1	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Figure 23. Numeric Bit Configurations, Decimal Digits 0-9, Seven-Bit Alphameric Code

Combinations of zone and numeric bits represent alphabetic and special characters. The B and A bits provide for three possible bit combinations: 10, 01, and 11. Figure 24 shows the zone and numeric bit combinations used to represent numeric, alphabetic, and special characters in the IBM 705 Data Processing System. In other systems using this code, there may be special characters not shown; however, these characters follow the same scheme of bit arrangement.

The C position, known as the check bit, is used for code checking only. Because the seven-bit alphameric code is an even parity code, the number of bits used to represent a character must have an even number of bits or the character is considered invalid. The check bit is present in a character when the sum of the zone and numeric bits used to represent the character is odd. If the number of bits in a character is even without the C bit, the C bit is not used.

Two-Out-of-Five Fixed Count Code

This code uses five positions of binary notation with the assigned values of 0, 1, 2, 3, and 6. In the basic code, decimal values are represented by bits present in only two of the five bit positions (Figure 25). The total number of possible combinations is ten–one for each decimal digit. The digits 1 through 9 are each composed of two bits, the position value sum of which equals the number to be represented. Zero is designated by the 1-2 bit combination.

Alphabetic and special characters are represented as a two-digit number. For example, the letter A is equal

	CHAR.	C	BA	8421	CHAR.	C	BA	8421	CHAR.	C	BA	8421	Storage Mark and Drum Mark	C	BA	8421
	&	0	11	0000	-	1	10	0000	Blank	1	01	0000	0	0	00	0000
ALPHABETIC	A	1	11	0001	J	0	10	0001	/	0	01	0001	CH. 1	1	00	0001
	B	1	11	0010	K	0	10	0010	S	0	01	0010	2	1	00	0010
	C	0	11	0011	L	1	10	0011	T	1	01	0011	3	0	00	0011
	D	1	11	0100	M	0	10	0100	U	0	01	0100	4	1	00	0100
	E	0	11	0101	N	1	10	0101	V	1	01	0101	5	0	00	0101
	F	0	11	0110	O	1	10	0110	W	1	01	0110	6	0	00	0110
	G	1	11	0111	P	0	10	0111	X	0	01	0111	7	1	00	0111
	H	1	11	1000	Q	0	10	1000	Y	0	01	1000	8	1	00	1000
	I	0	11	1001	R	1	10	1001	Z	1	01	1001	9	0	00	1001
	Plus Zero	0	11	1010	Minus Zero	0	1	10 1010	Record Mark		1	01 1010	Numerical Zero	0	0	00 1010
SPECIAL CHAR	.	1	11	1011	$	0	10	1011	,	0	01	1011	#	1	00	1011
	¤	0	11	1100	*	1	10	1100	%	1	01	1100	@	0	00	1100
	Group Mark	0	11	1111									Tape Mark	0	00	1111

NUMERICAL

Figure 24. IBM 705 Character Code Chart

to the coded decimal value of 61 and is composed of the two coded decimal digits of 6 and 1 (Figure 26).

Each digit that is moved in data processing operations is tested to assure that it has two bits, neither more nor less. This is a fixed count check.

Decimal Digit	Place Value 0	1	2	3	6
0	0	1	1	0	0
1	1	1	0	0	0
2	1	0	1	0	0
3	1	0	0	1	0
4	0	1	0	1	0
5	0	0	1	1	0
6	1	0	0	0	1
7	0	1	0	0	1
8	0	0	1	0	1
9	0	0	0	1	1

Figure 25. Two-Out-of-Five Code

One Digit					One Digit				
0	1	2	3	6	0	1	2	3	6
1	0	0	0	1	1	1	0	0	0

Figure 26. Letter A in Two-Out-of-Five Code

Bi-quinary Code

The bi-quinary code consists of seven positions of binary notation. Two positions, the binary positions, have assigned values of 0 and 5. The remaining positions, the quinary positions, have assigned values of 0, 1, 2, 3, and 4.

The digits 0 through 9 are coded by a combination of one binary bit and one quinary bit. The sum of the bit position values equals the number to be represented. Figure 27 illustrates the bit combinations used to code the digits 0 through 9.

Alphabetic and special characters are represented as two digits. For example, a letter A is equal to a decimal value of 61 and is composed of the coded decimal values, 6 and 1 (Figure 28).

Decimal Digit	Binary Position 0	5	Quinary Position 0	1	2	3	4
0	1	0	1	0	0	0	0
1	1	0	0	1	0	0	0
2	1	0	0	0	1	0	0
3	1	0	0	0	0	1	0
4	1	0	0	0	0	0	1
5	0	1	1	0	0	0	0
6	0	1	0	1	0	0	0
7	0	1	0	0	1	0	0
8	0	1	0	0	0	1	0
9	0	1	0	0	0	0	1

Figure 27. Bi-quinary Code

One Digit							One Digit						
0	5	0	1	2	3	4	0	5	0	1	2	3	4
0	1	0	1	0	0	0	1	0	0	1	0	0	0

Figure 28. Letter A in Bi-quinary Code

Each digit that is moved in the various processing operations is tested to assure that it is composed of one binary bit and one quinary bit only.

Six-Bit Numeric Code

In this code, six positions of binary notation are used. The positions are divided into three groups: one check bit, one flag bit, and four numeric bits with the assigned values of 8, 4, 2, and 1 (Figure 29). The decimal digits 0-9 are represented in binary coded decimal form, using the four numeric bit positions (Figure 30). Only bit combinations whose sum is 9 or less are used.

The F (flag) bit is used for special indications not related to the actual coding of the digits. For example, the presence or absence of a flag bit in the units digit of a numeric data field determines the sign (+ or −) of the field.

Check Bit	Flag Bit	Numeric Bit Positions			
C	F	8	4	2	1

Figure 29. Bit Positions, Six-Bit Numeric Code

Decimal Digit	Place Value C	F	8	4	2	1
0	1	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	1	0	0	0	1	1
4	0	0	0	1	0	0
5	1	0	0	1	0	1
6	1	0	0	1	1	0
7	0	0	0	1	1	1
8	0	0	1	0	0	0
9	1	0	1	0	0	1

Figure 30. Six-Bit Numeric Code

The C bit is used for parity checking. The six-bit numeric code is an odd parity code, so each digit must consist of an odd total number of bits, including C and F bits. The C bit is present only when an even total number of bits are in the numeric bit and F bit positions.

Alphabetic and special characters are represented as a special two-digit numeric value. For example, the letter A is equal to a decimal value of 41, and is composed of the coded decimal digits 4 and 1 (Figure 31).

One Digit							One Digit						
C	F	8	4	2	1	C	F	8	4	2	1		
0	0	1	0	0	0	0	0	0	0	0	1		

Figure 31. Letter A in Six-Bit Numeric Code

Binary System

Computers using this system of data representation are typified by the IBM 704, 709, and 7090 Data Processing Systems.

For these systems, the basic unit of information is the word. A word consists of 36 consecutive bit positions of information which are interpreted as a unit, much as a character or a digit in other systems (Figure 32).

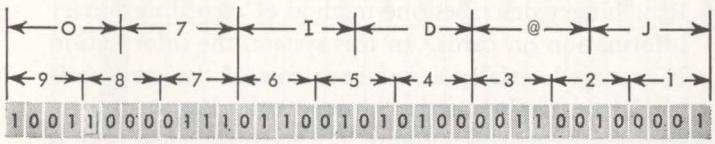


Figure 32. The Binary System

The bit positions within the word have a place significance related to the binary number system. That is, the place position of a bit in the word determines the value of the bit. In the binary number system, the decimal values of the places (from right to left) are 0, 1, 2, 4, 8, 16, 32, 64, and so on (Figure 32).

Although the place values of the bits of a word are always that of the binary number system, they can be interpreted or processed in such a way as to represent other than a binary number. For example, a 36-bit word (Figure 32) can be interpreted as one 36-place binary number, as a 9-digit decimal number, as six alphameric characters, or as any predetermined representation established by the programmer.

Data Recording Media

IBM Cards

The IBM punched card is one of the most successful media for communication with machines. Information is recorded as small rectangular holes punched in specific locations in a standard size card (Figure 33). Information, represented (coded) by the presence or absence of holes in specific locations, can be read or sensed as the card is moved through a card reading machine.

Reading or sensing the card is basically a process of automatically converting data recorded as holes to an electronic language and entering the data into the machine. Cards are used both for entering data into a machine and for recording or punching information from a machine. Thus, the IBM card is not only a means of transferring data from some original source to a machine, but also is a common medium for the exchange of information between machines.

IBM cards provide 80 vertical columns with twelve punching positions in each column. The twelve punching positions form twelve horizontal rows across the card. One or more punches in a single column represents a character. The number of columns used depends on the amount of data to be represented.

The card is often called a unit record because the data are restricted to the 80 columns and the card is read or punched as a unit of information. The actual data on the card, however, may consist of part of a record, one record, or more than one record. If more

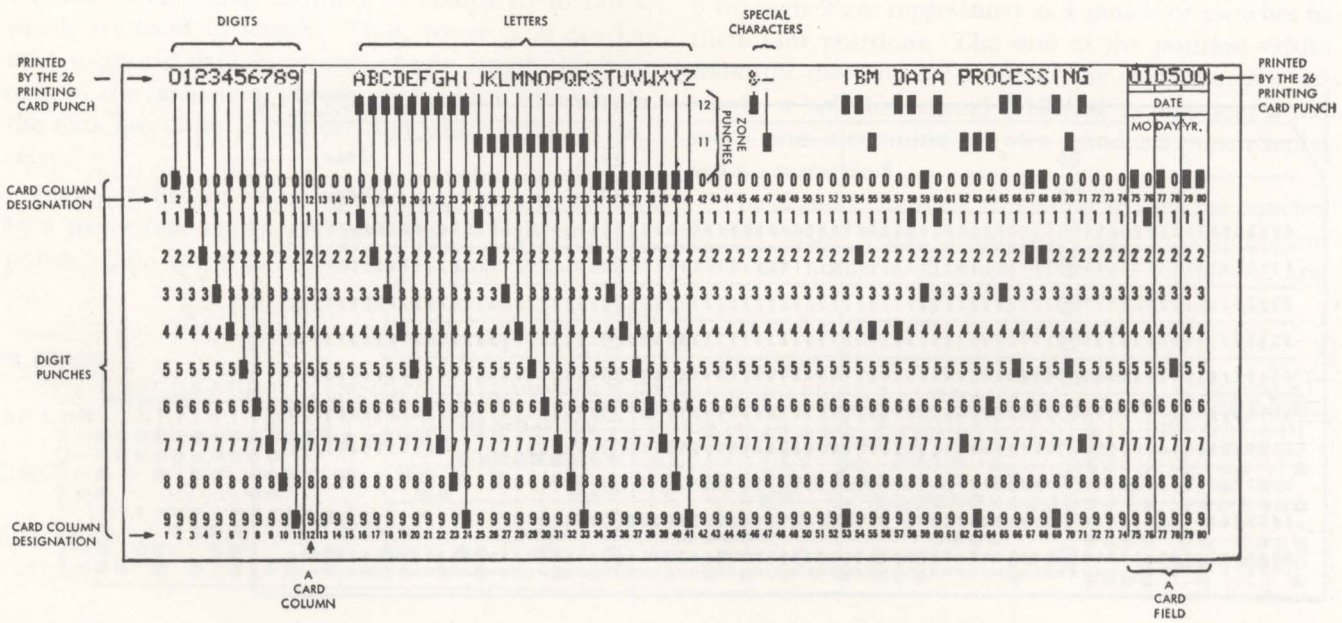


Figure 33. IBM Card — Standard Card Code

than 80 columns are needed to contain the data of a record, two or more cards may be used. Continuity between the cards of one record may be established by punching identifying information in designated columns of each card.

Information punched in cards is read or interpreted by a machine called a card reader and is recorded (punched) in a card by a machine called a card punch. Data are transcribed from source documents to punched cards by manually operated card punch machines.

IBM Card Code

The standard IBM card code uses the twelve possible punching positions of a vertical column on a card to represent a numeric, alphabetic, or special character (Figure 33). The twelve punching positions are divided into two areas: numeric and zone. The first nine punching positions from the bottom edge of the card are the numeric punching positions and have an assigned value of 9, 8, 7, 6, 5, 4, 3, 2, and 1, respectively. The remaining three positions 0, 11, and 12 are the zone positions. (The 0 position is considered to be both a numeric and zone position.)

The numeric characters 0 through 9 are represented by a single punch in a vertical column. For example, 0 is represented by a single punch in the 0 zone position of the column.

The alphabetic characters are represented by two punches in a single vertical column: one numeric punch and one zone punch. The alphabetic characters A through I use the 12 zone punch and a numeric punch 1 through 9, respectively. The alphabetic characters J through R use the 11 zone punch and a numeric punch 1 through 9, respectively. The alphabetic characters S through Z use the 0 zone punch and a numeric punch 2 through 9, respectively.

The standard special characters $ * % and so on, are represented by one, two, or three punches in a column of the card and consist of punch configurations not used to represent numeric or alphabetic characters.

Row Binary Data Representation

Row binary describes one method of recording binary information on cards. In this system, the information is arranged serially across each row of the card (left to right) starting at the 9-row, continuing to, and including the 12-row (Figure 34). Each punched hole is regarded as a binary 1. No punch indicates a binary 0.

In the IBM 704, 709, and 7090 systems, 72 columns of the card are used. Binary information is represented as follows: each of the 12 rows of the card is split into two parts; the left half consists of columns 1 through 36, and the right half consists of columns 37 through 72. One full word of binary information (36 bits) can be punched in any half row; 24 words may be contained on a card.

Column Binary Data Representation

Binary information may also be recorded in a columnar binary fashion. With this method, data are arranged in parallel with each column of the card containing 12 information bits. Thus, one full 36-bit word for the IBM 704, 709, or 7090 systems would require three card columns. The entire card could contain twenty-six 36-bit words (Figure 35).

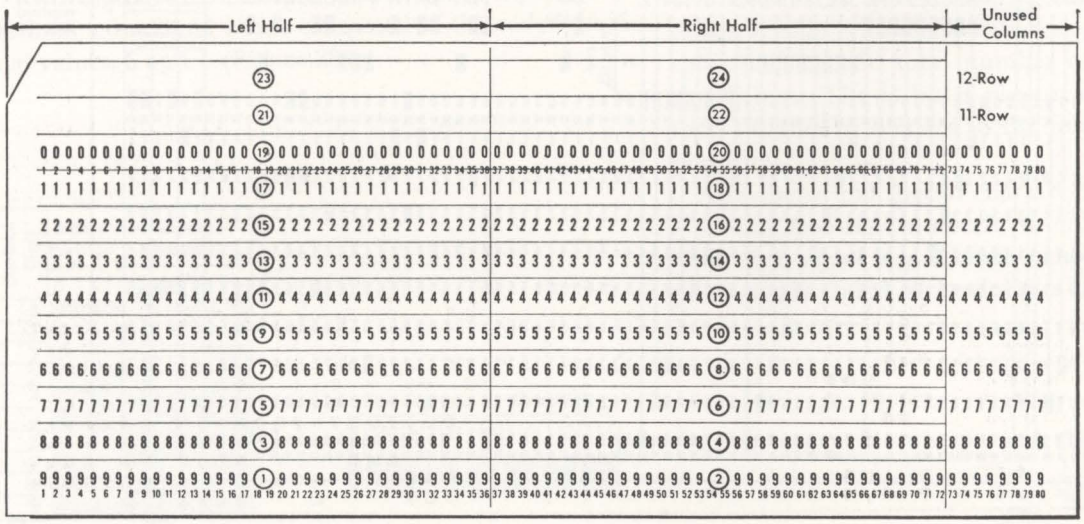


Figure 34. IBM Card — Row Binary

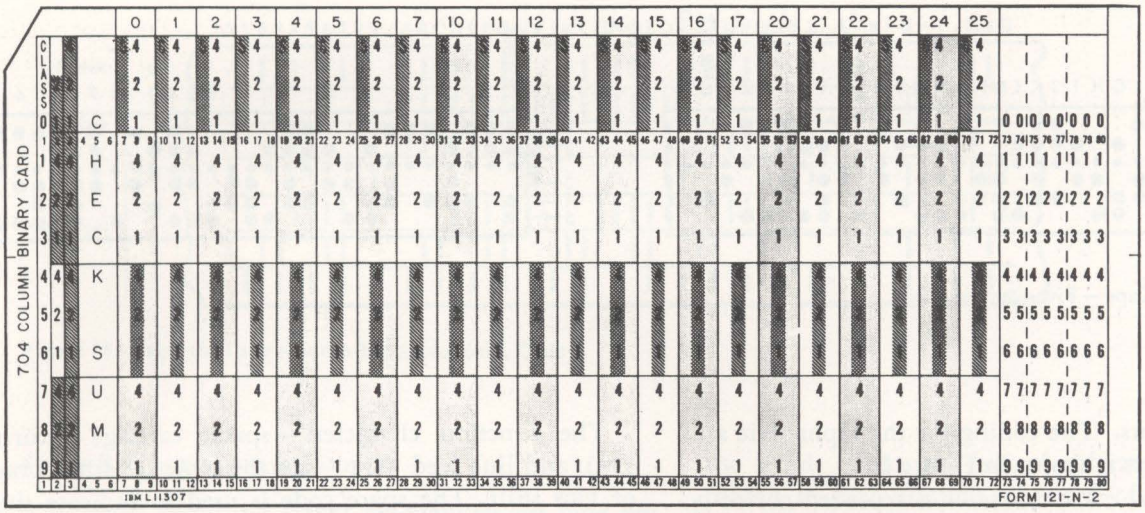


Figure 35. IBM Card — Column Binary

Paper Tape

Punched paper tape serves much the same purpose as punched cards. Developed for transmitting telegraph messages over wires between two machines, paper tape is now used for communication with other machines as well. For long distance transmission of data, machines convert data from IBM cards to paper tape, send the information over telephone or telegraph wires to produce a duplicate paper tape at the other end of the wire, and reconvert the information to punched cards (Figure 16).

Data are recorded as a special arrangement of punched holes, precisely arranged along the length of a paper tape (Figures 36 and 37). Paper tape is a continuous recording medium, as compared to cards which are fixed in length. Thus, paper tape can be used to record data in records of any length, limited only by the capacity of the storage medium into which the data are to be placed or from which data are received.

Data punched in paper tape are read or interpreted by a paper tape reader and recorded by a paper tape punch. Data are transcribed from source documents to paper tape by manually operated tape punching devices.

Eight-Channel Code

Data are recorded (punched) and read as holes located in eight parallel channels along the length of the paper tape. One column of the eight possible punching positions (one for each channel) across the width of the tape is used to code numeric, alphabetic, special, and function characters. Figure 36 shows a section of paper tape illustrating the eight channels and several coded characters.

The lower four channels of the tape (excluding the feed holes) are labeled 1, 2, 4, and 8 and are used to record numeric characters. The numeric characters 0 through 9 are represented as a punch or punches in these four positions. The sum of the position values indicates the numeric value of the character. For example, a hole in channel 1 is used to represent a numeric one; a combination of a 1 and a 2 punch represents a numeric 3.

The X and 0 channels are similar to the zone punches in IBM cards. These channels are used in combination with the numeric channels to record alphabetic and

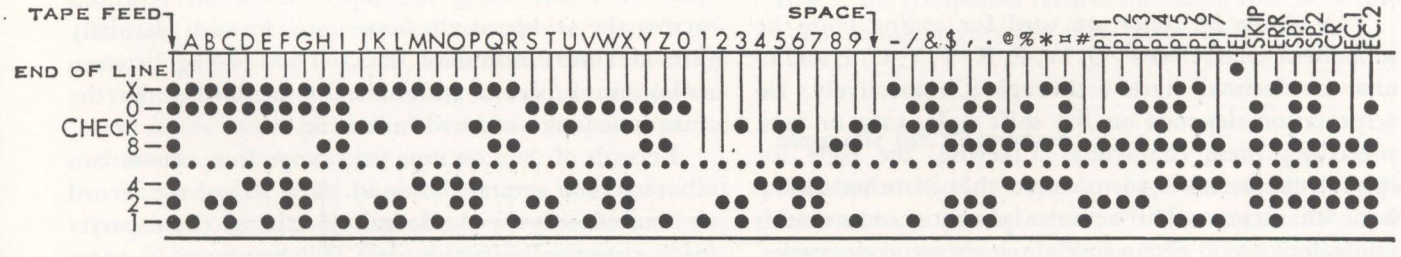


Figure 36. Paper Tape — Eight-Channel Code

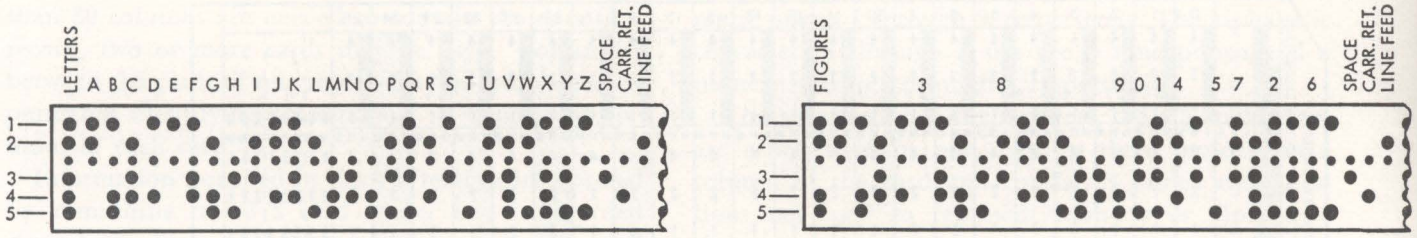


Figure 37. Paper Tape — Five-Channel Code

special characters. The coding for the alphabetic and special characters is shown in Figure 36.

To check that each character is recorded correctly, each column of the tape is punched with an odd number of holes. A check hole must be present in any column whose basic code (X, 0, 8, 4, 2, 1) consists of an even number of holes.

A punch in the EL channel is a special function character used to mark the end of a record on the tape. The tape feed code consists of punches in the X, 0, 8, 4, 2, and 1 channels and is used to indicate blank character positions. The paper tape reader automatically skips over areas of tape punched with the tape feed code.

Five-Channel Code

Data are recorded (punched) and read as holes in five parallel channels along the length of the paper tape. One column of the five possible punching positions (one for each channel) across the width of the tape is used to code numeric, alphabetic, special, and function characters. Figure 37 shows a section of paper tape illustrating the five channels and several coded characters.

Because there are only 31 possible combinations of punches, using the five punching positions, a shift system is used to double the number of available codes. When the letters (LTRS) code punch precedes a section of tape, the characters that follow are interpreted as alphabetic characters (Figure 37). When the figures (FIGS) code punch precedes a section of tape, the coded punches are interpreted as numeric or special characters.

Ten of the 31 codes are used for coding both the alphabetic characters P, Q, W, E, R, T, Y, U, I, and O and the decimal digits 0 through 9, respectively. Interpretation depends on the shift code, LTRS or FIGS, preceding these characters. Likewise, the code for special characters is identical to that of other alphabetic characters. The actual alphabetic code that is equivalent to a given special character code varies, depending on customer requirements.

The function characters — space, carriage return (CR), and line feed (LF) — are the same in either LTRS or FIGS shift. The space code is used to indicate the absence of data on tape. The actual function of the CR and LF characters depends on the machine with which they are used.

Magnetic Tape

Magnetic tape is the most recently developed medium for recording data for machine processing; it is the principal input-output medium used by computer systems.

IBM magnetic tape is similar to the tape used in home tape recorders. It is a plastic tape, one-half inch wide, and coated on one side with a metallic oxide. Data are recorded as magnetized spots or bits in the metallic oxide (Figures 38 and 39). Information recorded on tape is permanent and can be retained for an indefinite time. Previous recordings are destroyed as new information is written. This means that tape can be used repetitively with significant savings in recording costs. Several types of magnetic tape are available to meet varying requirements of strength, durability, reliability and cost.

For handling and processing, tape is wound on plastic reels containing up to 2400 feet of tape. (Lengths as short as 50 feet may be used.) The magnetic tape unit, which functions both as an input and output device, moves the magnetic tape and accomplishes the actual reading or writing of information on the tape. Data are recorded in seven parallel channels or tracks along the tape. Seven bit positions across the width of the tape (one in each channel) provide one column of data. The spacing between columns of bits is automatically established by the magnetic tape unit used in writing.

Records of data on tape may range from one or two characters to several thousand. The size of the record is limited only by the length of tape or the capacity of the storage units that data will be placed in or removed from.

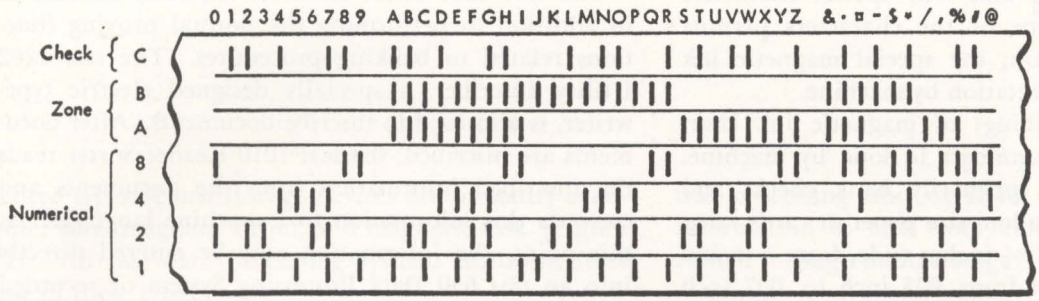


Figure 38. Magnetic Tape — Seven-Bit Alphameric Code

Seven-Bit Alphameric Code

The seven recording tracks or channels on tape are labeled C, B, A, 8, 4, 2, 1 and correspond to the seven bit positions of the seven-bit alphameric code. A character is represented by the presence or absence of bits in the seven channel positions of one column, across the width of the tape. Figure 38 shows characters in the seven-bit alphameric code as they appear on tape.

To verify tape reading and writing, each character is checked for even parity. In addition to this vertical parity check, a horizontal (longitudinal) parity check is made on each record. At the time a record is written, the bits in each horizontal row are counted. At the end of the record, a check character is recorded. This character has a bit corresponding to each channel row with an odd bit count. Thus, when the record is read, each channel row of the complete record, including the check character, should satisfy the even parity condition. The check character serves this purpose only, and is never included as part of the record when data are transferred to the computer system.

Tape written in the seven-bit alphameric code can be used by several data processing systems, providing a means of intercommunication from one system to another. There are instances, however, where special characters, peculiar to only one system, are written on tape. For this reason, consideration must be given to the characters used when tape written on one system may be used on another.

Binary System

Binary information recorded on tape is related primarily to the IBM 704, 709, and 7090 Data Processing Systems. With these systems the basic unit of information is the word—36 consecutive bits—compared to the character or digit of other systems.

To record a word of data on tape, the seven bit positions of each column on tape are used; however, the C bit position of the column is for parity checking purposes only, and is not considered a part of the word. Thus, six bits of information can be recorded in each column. A word of 36 bits is represented in six consecutive columns on tape (Figure 39).

To verify accuracy of tape reading and writing, each column of bits must consist of an odd number of bits and is tested to insure odd parity. As tape is written, check bits are automatically added to the columns that have an even number of bits. In addition to this vertical parity check, a horizontal (longitudinal) parity check is made on each record. At the time a record is written, the bits of each horizontal row are counted.

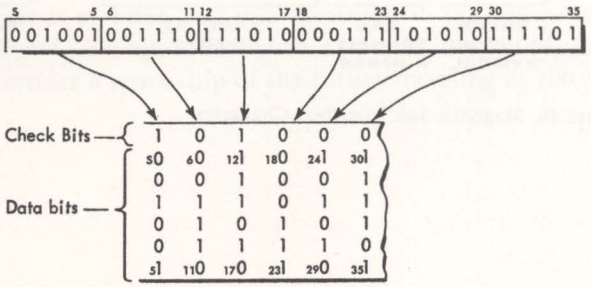


Figure 39. Magnetic Tape — Binary System

At the end of the record, a check character is recorded. This character has a bit corresponding to each row with an odd bit count. When the record is read, each row of the completed record, including the check character, should satisfy the even parity condition.

Magnetic Ink Characters

Another method of representing data on paper media for machine processing is with magnetic ink characters—a language readable by both man and machine. Magnetic ink characters are printed on paper as the

arabic numerals 0 to 9 and four special characters (Figure 40). The shape of the characters permits easy visual interpretation; the special magnetic ink allows reading or interpretation by machine.

The printing (inscribing) of magnetic ink characters on the paper documents is done by machine. The paper documents, primarily bank checks and deposit slips, may be random size paper or cards ranging from 2¾ inches to 3⅔ inches wide, from 6 inches to 8¾ inches long, and from .003 inch to .007 inch thick.

The IBM 1201 Proof Inscriber inscribes documents in addition to performing the normal proving functions related to banking procedures. The IBM 1202 Utility Inscriber, a specially designed electric typewriter, is also used to inscribe documents. After documents are inscribed, the IBM 1210 Reader Sorter reads the inscribed information from the documents and converts this information to a machine language. At this point, the information may be entered directly into an IBM 650 Data Processing System or recorded on magnetic tape as input to other systems.

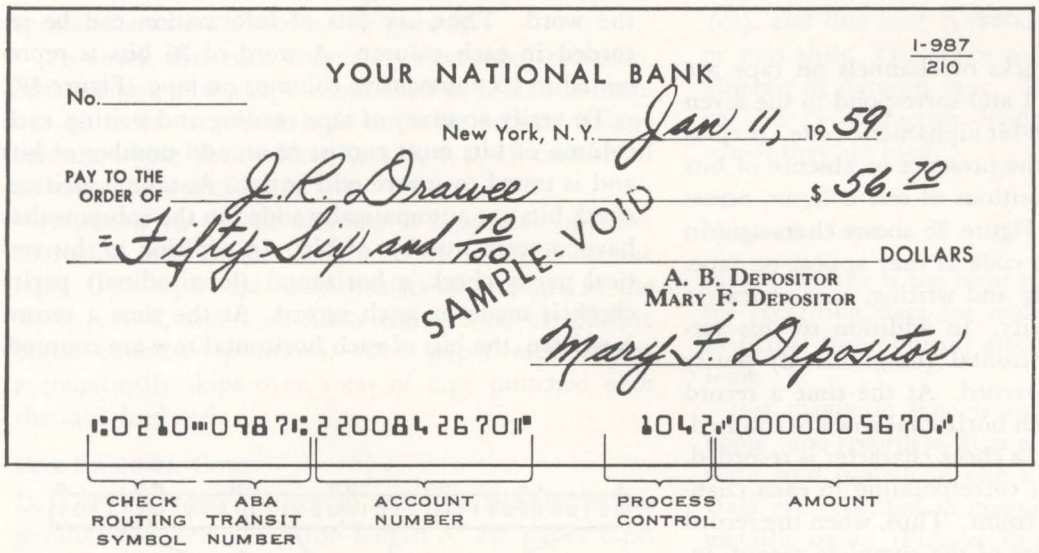


Figure 40. Magnetic Ink Inscribed Characters

Three types of IBM storage devices are presently available: core, magnetic drum, and magnetic disk (Figure 41). All IBM data processing systems utilize at least one of these types.

Information can be placed into, held in, or removed from computer storage as needed. The information can be:

1. Instructions to direct the central processing unit.
2. Data (input, in-process, or output).
3. Reference data associated with processing (tables, code charts, constant factors, and so on).

Storage is classified as either main or auxiliary (Figure 42).

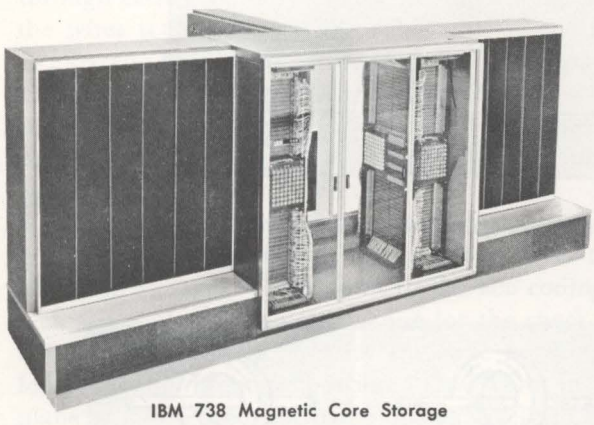
Main or primary storage accepts data from an input unit, exchanges data with and supplies instructions to the central processing unit, and can furnish data to an output unit. All data to be processed by any system must pass through main storage. This unit must therefore have capacity to retain a usable amount of data and the necessary instructions for processing.

An occasional application can require additional storage. In this instance the capacity of main storage is augmented by an auxiliary or secondary storage unit. Auxiliary storage is not directly accessible to the central processing unit or input or output devices; all information to and from auxiliary storage must be routed through main storage.

Storage is arranged somewhat like a group of numbered mail boxes in a Post Office (Figure 43). Each box is identified and located by its number. In the same way, storage is divided into locations, each with an assigned address. Each location holds a specific unit of data. Depending on the system, the unit of data may be a character, a digit, a complete record, or a word. To insert or remove data at a location, the address must be known.

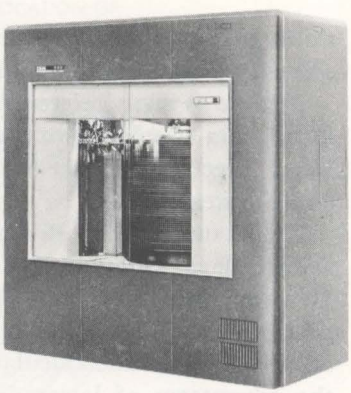
When information enters a location, it replaces the previous contents of that location. However, when information is taken *from* a location, the contents remain unaltered. Thus, once located in storage, the same data may be used many times. In effect, a duplicate of the information is made available for processing.

The computer requires some time to locate and transfer information to or from storage. This is called access time. Storage units are available where access time is so brief that it is measured in millionths of a second. To appreciate such a minute interval of time, consider a space-ship of the future traveling at 100,000



IBM 738 Magnetic Core Storage

IBM 733 Magnetic Drum Storage



IBM 355 Disk Storage

Figure 41. IBM Storage Devices

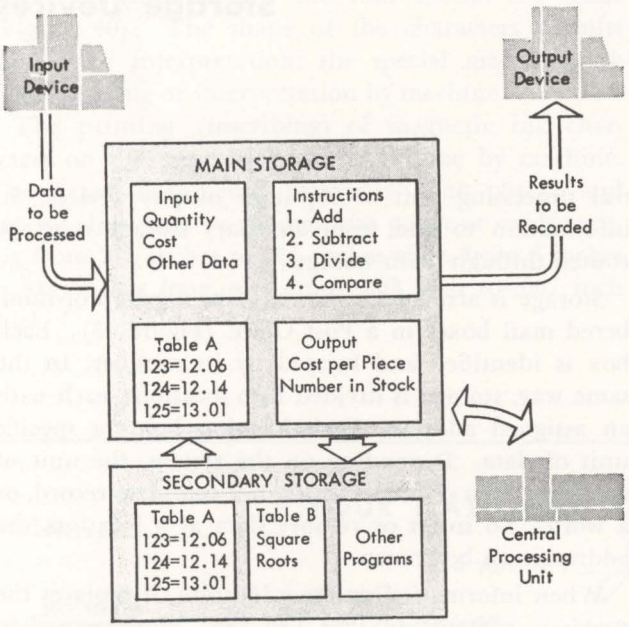


Figure 42. Schematic, Main and Secondary Storage

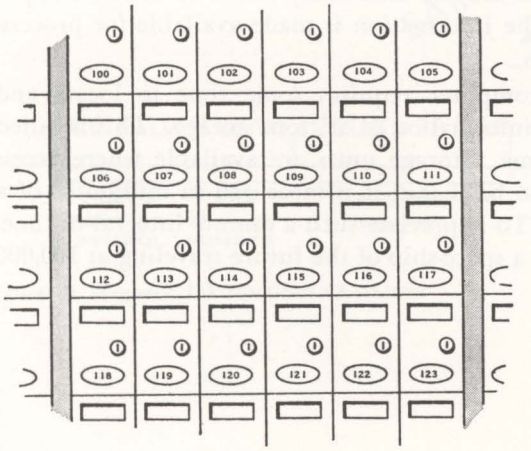


Figure 43. Post Office Mail Boxes

miles per hour. In one millionth of a second, the space-ship would travel approximately 1¾ inches.

Because so many references must be made to storage in all data processing operations, the speed of access has a direct bearing on the efficiency and cost of the entire system.

For example, core storage is the most expensive storage device in terms of cost per storage location. However, core storage also provides the fastest access time and, thus, may be the most economical in terms of cost per machine calculation. Drum storage offers the advantages of lower direct cost to offset slower speed. Most disk storage devices are slower than drum storage but offer the advantage of capacity in millions of digits.

Core Storage

A magnetic core is a tiny ring of ferromagnetic material, a few hundredths of an inch in diameter. Each core is pressed from a mixture of ferric oxide powder and other materials and then baked in an oven.

Aside from its compact size – a decided advantage in computer design – the important characteristic of the core is that it can be easily magnetized in a few millionths of a second. And, unless deliberately changed, it retains its magnetism indefinitely.

If cores are placed on a wire like beads on a string and a strong enough electrical current is sent through the wire, the cores become magnetized (Figure 44). The direction of current determines the polarity or magnetic state of the core (Figure 45). Reversing the direction of current changes the magnetic state (Figure 46). Consequently, the two states can be used to represent 0 or 1, plus or minus, yes or no, or on or off conditions. For machine purposes, this is the basis of the binary system of storing information.

More than one million cores are used in a large IBM core storage. Because any specified location of storage must be instantly accessible, the cores are arranged so that any combination of ones and zeros representing a character can be written magnetically or "read" back when needed.

To accomplish selection, two wires run through each core at right angles to each other (Figure 47). When

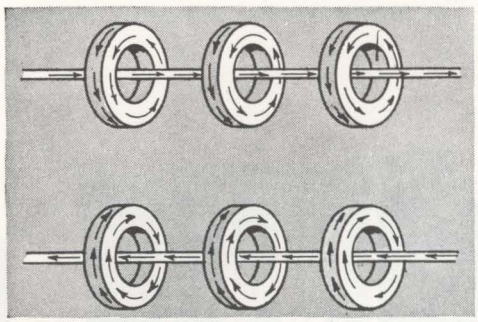


Figure 44. Polarity of Magnetic Cores

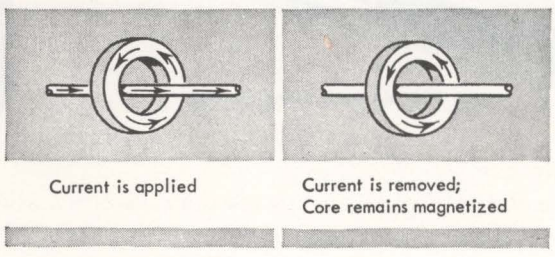


Figure 45. Magnetizing a Core

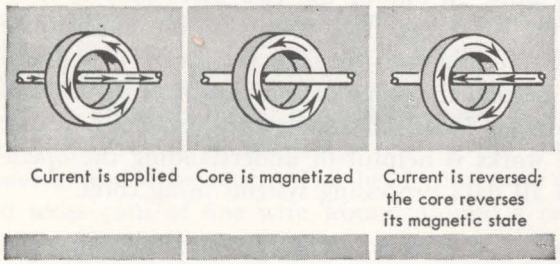


Figure 46. Reversing a Core

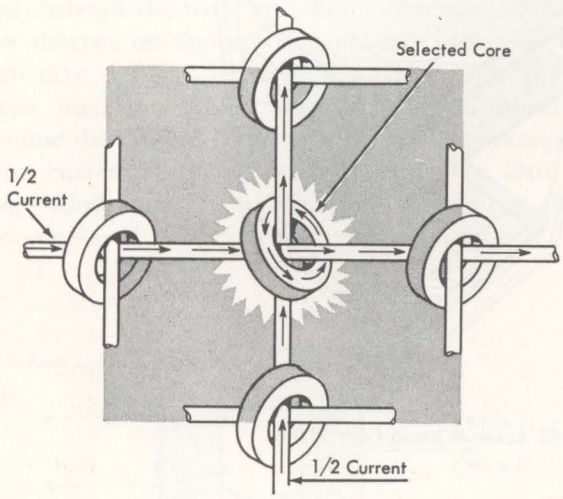


Figure 47. Selecting a Core

half the current needed to magnetize a core is sent through each wire, only the core at the intersection of the wires is magnetized. No other core in the string is affected. Using this principle, a large number of cores can be strung on a screen of wires; yet, any single core in the screen can be selected for storage or reading without affecting any other. Such an assembly of wires is called a plane (Figure 48).

To illustrate the use of a number of planes to store a BCD character, assume that the letter A is to be placed in storage. To conform to the BCD coding system, seven planes are needed: one for the check position of the character, two for the zone portion, and four for the numeric portion. One core in each plane is magnetized positively or negatively to represent the binary configuration for the letter A, 1 11 0001 (Figure 49).

Note that the planes are stacked in an array, with all cores representing A at the intersection of the same two wires in each plane. If an imaginary line were drawn vertically through the cores representing A, the line would show the physical location of one character position of storage.

In a typical storage unit (IBM 705) each plane is 50 cores wide by 80 cores long, making a total of 4,000 per plane. An array of 35 such planes provides capacity for 140,000 bits. An imaginary vertical line drawn through this array shows five positions. (It passes through 35 cores, seven cores per character, consequently five positions, as shown in Figure 50.) Thus, the capacity of this unit is 20,000 characters.

Once information is placed in core storage, some means must be devised to make it accessible, that is, to recall it when needed. It has been shown that a definite magnetic polarity can be set up in a core by the flow of current through a wire. In the machine, the flow is not actually constant; it is sent through the wire as an electrical pulse. This pulse is said to flip the core to a positive or negative state, depending on the direction of current flow.

If the magnetic state of the core is reversed by the pulse, this abrupt change or flip induces current in a

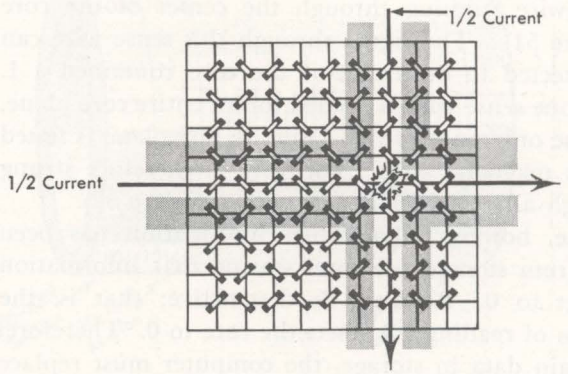


Figure 48. Magnetic Core Plane

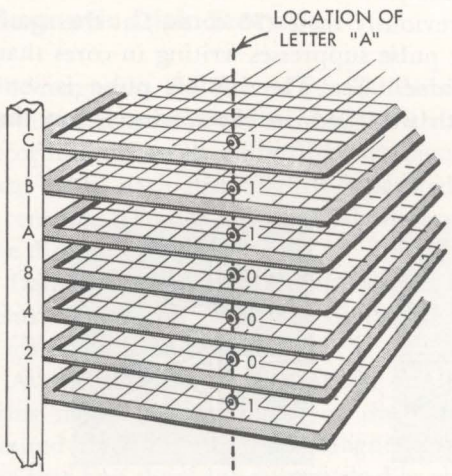


Figure 49. BCD Character Location

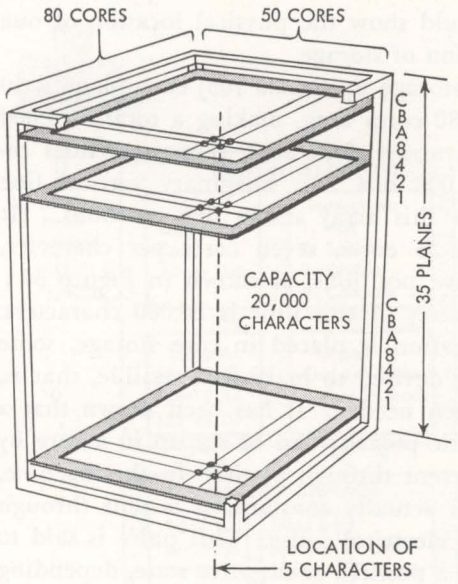


Figure 50. Schematic, 20,000 Position Storage

third wire running through the center of the core (Figure 51). The signal through this sense wire can be detected to determine if the core contained a 1. Only one sense wire is needed for an entire core plane, because only one core at a time in any plane is tested for its magnetic state. The wire is therefore strung through all the cores of the plane (Figure 52).

Note, however, that when information has been read from storage, all cores storing that information are set to 0. Read-out is destructive; that is, the process of reading a 1 resets the core to 0. Therefore, to retain data in storage, the computer must replace 1's in those cores that had previously contained 1's. But cores that contained 0's must remain as 0's.

To reproduce (regenerate) the 0's and 1's as they should be, the computer tries to write back 1's in all the locations previously read (35 cores); at the same time, an inhibit pulse suppresses writing in cores that previously contained 0's. The inhibit pulse is sent through a fourth wire and, in effect, cancels out the writing pulse in one of the two wires used to magnetize the core. Like the sense wire, the inhibit wire (Figure 53) also runs through every core in a plane.

It is beyond the scope of this book to fully explain core storage. However, a basic knowledge of how core storage works is helpful in understanding the operation of all data processing systems using cores.

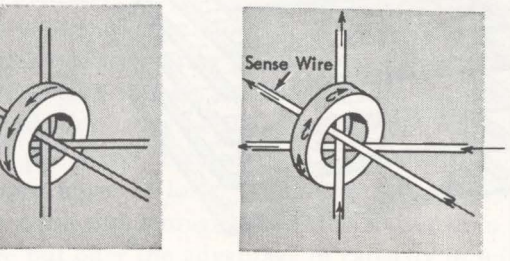


Figure 51. Core Sense Wires

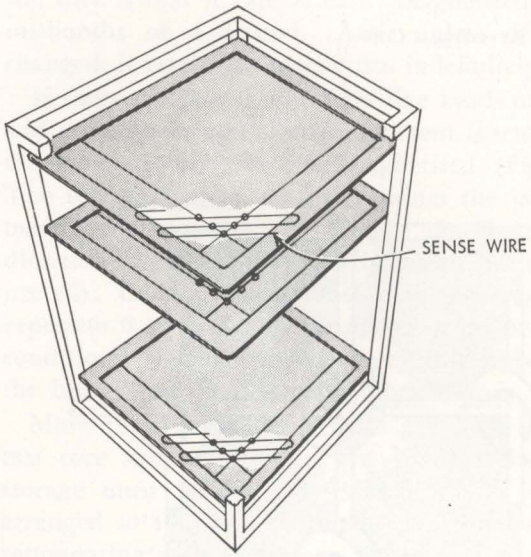


Figure 52. Sense Wire in Core Plane

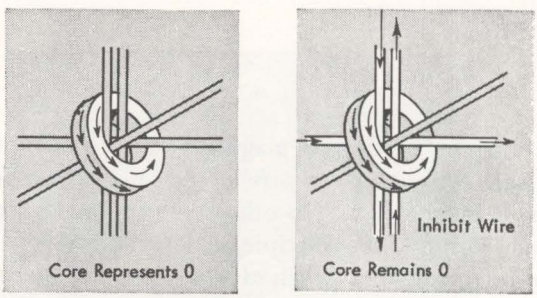


Figure 53. Core Inhibit Wire

Magnetic Drum Storage

A magnetic drum is a steel cylinder enclosed in a copper sleeve. The copper surface is plated with a cobalt and nickel alloy. This coating of the surface of the drum is the actual storage medium.

If an area of this material is placed in a magnetic field, it becomes magnetized. After the magnetizing force is removed, the magnetism is retained indefinitely. The area affected can be quite small (on one model, it is about .071 inches long) so that a large

number of magnetized spots or *cells* can be placed in a small space. The effect of magnetizing a cell is the same as if a tiny bar magnet were imbedded in the surface of the drum.

As the drum rotates at a constant speed, information is written by magnetizing cells as the surface passes a read-write head. The head consists of read and write coils of fine wire wound around a center core. A plastic shim separates the ends of the core, providing a magnetic gap. The head assembly is positioned close to the drum so that magnetic lines of force produced by the write head fringe around the gap and flow through the alloy surface (Figure 54).

The cells are magnetized by sending pulses of current through the write coil. The direction of current flow determines the resulting polarity of a cell. Consequently, cells can represent either 1's or 0's, the two digits used for binary recording in all machines. Because the drum is rotating while writing takes place, write current must be extremely short to limit the magnetized area. Thus, the size of the cell is almost the same as if the drum were motionless.

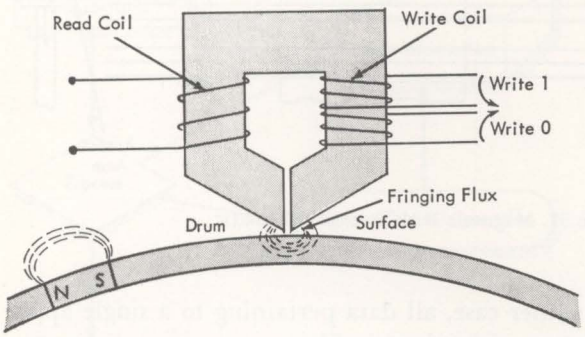


Figure 54. Drum Recording

When a cell that has been magnetized passes under the read-write head, its magnetic state can be sensed by current induced in the read coil. In this way, information written on the drum can be read back when needed. Reading is not destructive because the condition of a cell is not changed as it passes the head. Unlike core storage, the drum needs no regeneration process, and the information can be read again and again without being erased. Drum storage is, therefore, permanent and data on its surface remain there indefinitely even after the power to a system is turned off. Information is replaced only when new information is written.

Each drum has a specific number of storage locations, each of which is addressable by the computer. The capacity of each location depends upon the design of the drum and the data representation used (Figure 55).

Because reading or writing can occur only when a specified location is passing the heads, access time may vary, depending upon the distance to be traveled by that location to the head.

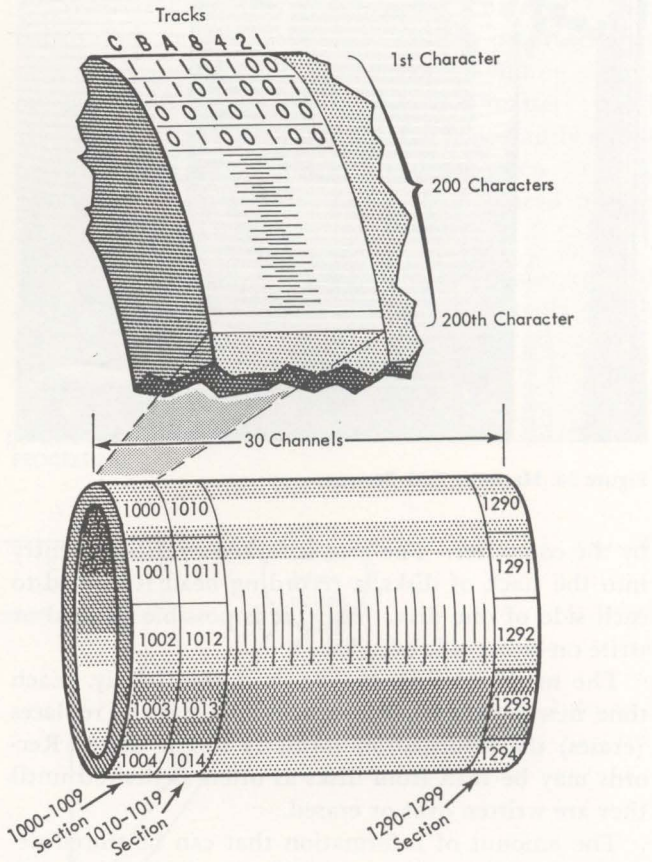


Figure 55. Schematic, Drum Storage of IBM 705 Data Processing System

Magnetic Disk Storage

The magnetic disk is a thin metal disk about two feet in diameter, coated on both sides with a ferrous oxide recording material. Fifty disks are mounted on a vertical shaft, each disk slightly separated from the adjacent one (Figure 56). The shaft revolves, spinning the disks at 1200 RPM.

Data are stored as magnetized spots located in concentric tracks on each face of the disk (Figure 57). Each track is addressable.

At the side of the disk stack, one or more access arms move under control of the computer to any desired track on any disk. Magnetic recording heads mounted on these access arms read or write as directed

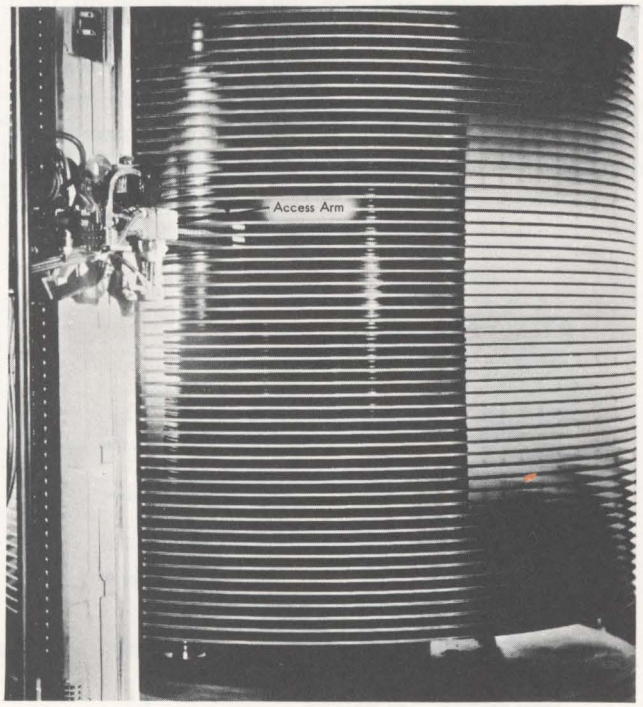


Figure 56. Magnetic Disk Storage

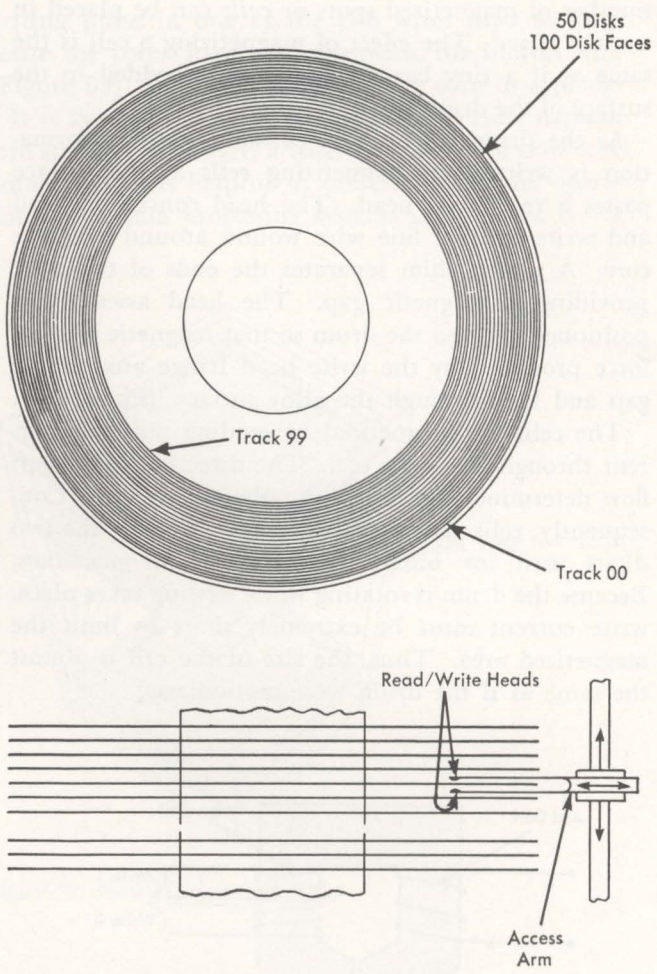


Figure 57. Magnetic Disk Storage Schematic

by the computer. The arm is forked so that, on entry into the stack of disks, a recording head is carried to each side of one disk. Thus, it is possible to read or write on either side of a disk.

The magnetic disk can be used repetitively. Each time new information is stored in a track it replaces (erases) the information formerly stored there. Records may be read from disks as often as desired until they are written over or erased.

The amount of information that can be stored depends on the number of disks, the number of disk tracks on each face, and the method of coding the stored information.

Access time is the total time required to locate the proper disk and the specific track on the disk and for the addressed location to pass under the recording head. When more than one access arm is used, the effective access time can be reduced; while one arm is reading or writing, a second access arm can locate the next storage position.

Storage and Data Processing Methods

IBM data processing systems use two methods of data handling: sequential, or batch processing, and in-line, or random access processing. (See Figure 58.) The organization and capacity of the main storage device determine which method applies.

In either case, all data pertaining to a single application are maintained in files.

In sequential processing, these files are stored outside the computer — usually on magnetic tape — and they are arranged in a predetermined sequence. The data may concern inventory, accounts receivable, accounts payable, payroll, and the like. Each file is made up of records, each containing information required to describe completely a single item. The sequence may be by item number, name, account number, or man number, but all files pertaining to a single application must be in the same sequence.

In many cases, processing involves not only performing calculation on some parts of each record to arrive at balances, amounts, or earnings, but also involves adding, changing, or deleting records as new transactions occur. However, before transactions can be applied against the main or master file, they must also be arranged in the same sequence as the master file. For this purpose, they are accumulated in convenient groups or batches.

The two files, master and transaction, now become input to the data processing system. One record or a small group of records is read into storage at a time. These are processed and the result is written as output. The next group of records is read in, and the process is repeated. The series of repetitive operations continues under direction of program instructions, record by record, until the input files are exhausted. The results form a revised master file, updated according to the current transactions. The new master file is in the same sequence as the original files.

Other output may also be produced as a by-product of the processing. This output may be records of delinquent accounts, bank orders, earnings statements, payroll checks, and so on. In every case, however, the sequence of all output remains the same as the sequence of the incoming data.

With sequential processing, the information in storage is transient. Consequently, the storage unit needs only enough capacity for program instructions plus the largest element of data to be processed.

When in-line processing is used, all information concerning application status is held in a large-capacity storage unit, usually a magnetic disk file. Storage of information is permanent and data can be retained indefinitely.

Transactions affecting the contents of the file are fed to the computer at random, as they occur. In this case, the computer locates the corresponding record or data in storage and adjusts this master record accordingly. Accounts or balances are constantly maintained and are available as output when needed. Transactions are not batched and they need not be sorted before processing.

Other output can also be obtained under control of the stored program, or the entire contents of the disk file can be written out as required.

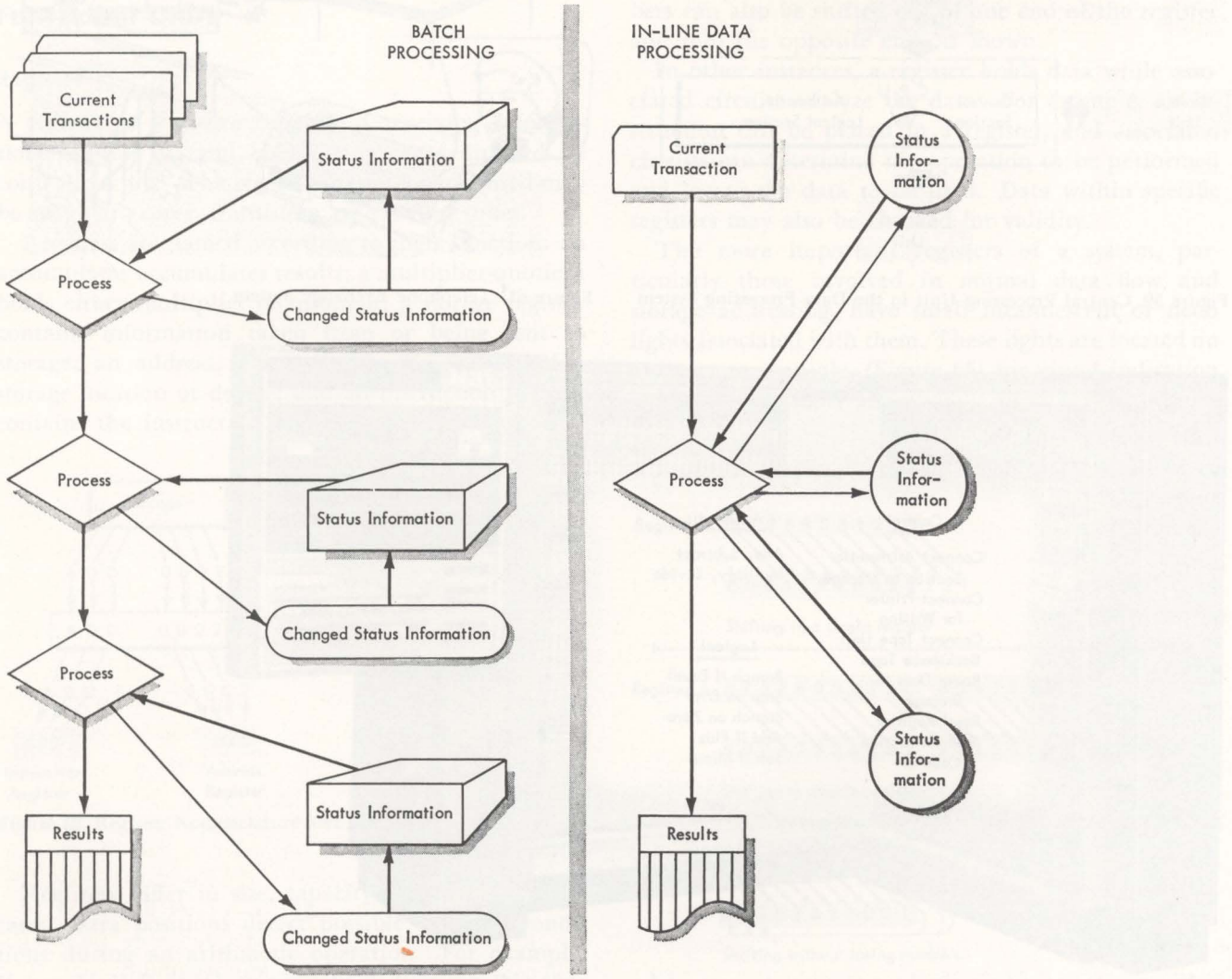


Figure 58. Batch and In-line Data Processing

Central Processing Unit (CPU)

The central processing unit controls and supervises the entire computer system and performs the actual arithmetic and logical operations on data. From a functional viewpoint, the central processing unit consists of two sections: control and arithmetic-logical (Figure 59).

The control section directs and coordinates all operations called for by instructions. This involves control of input-output devices, entry or removal of information from storage, and routing of information between storage and the arithmetic-logical section (Figure 60). Through the action of the control section, automatic, integrated operation of the entire computer system is achieved.

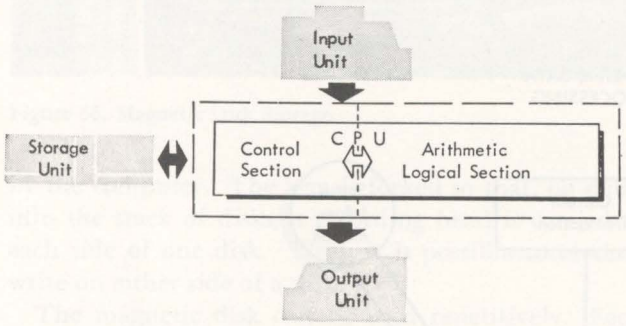


Figure 59. Central Processing Unit in the Data Processing System

In many ways, the control section can be compared to a telephone exchange. All possible data transfer paths already exist, just as there are connecting lines between all telephones serviced by a central exchange (Figure 61).

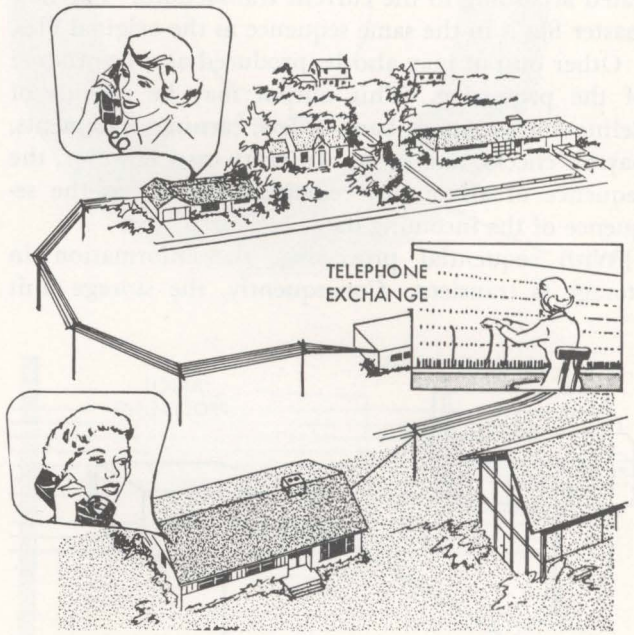


Figure 61. Telephone Exchange System

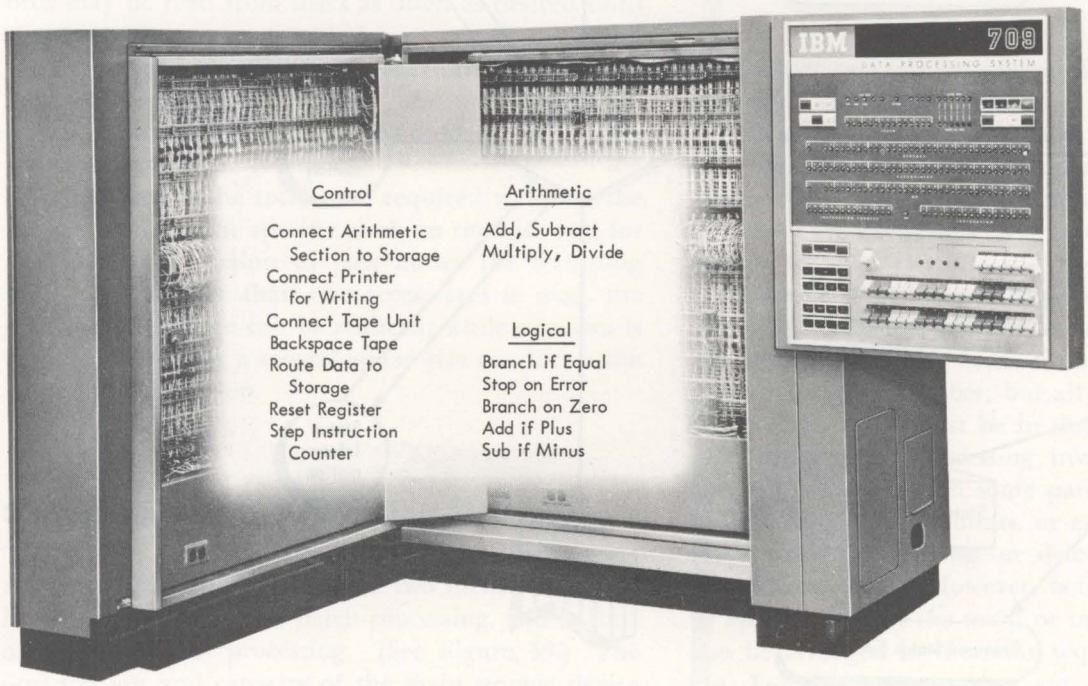


Figure 60. Control and Arithmetic-Logical Sections

The telephone exchange has a means of controlling instruments that carry sound pulses from one phone to another, ring the phones, connect and disconnect circuits, and so on. The path of conversation between one telephone and another is set up by appropriate controls in the exchange itself. In the computer, execution of an instruction involves opening and closing many paths or *gates* for a given operation. The control section can start or stop an input-output unit, turn a signal device on or off, rewind a tape reel, or direct some process of calculation.

The arithmetic-logical section contains the circuitry to perform arithmetic and logical operations. The arithmetic portion calculates, shifts numbers, sets the algebraic sign of results, rounds, compares, and so on. The logical portion carries out the decision-making operations to change the sequence of instruction execution.

Functional Units

Register

A register is a device capable of receiving information, holding it, and transferring it as directed by control circuits. The electronic components used may be magnetic cores, transistors, or vacuum tubes.

Registers are named according to their function: an accumulator accumulates results; a multiplier-quotient holds either multiplier or quotient; a storage register contains information taken from or being sent to storage; an address register holds the address of a storage location or device; and an instruction register contains the instruction being executed (Figure 62).

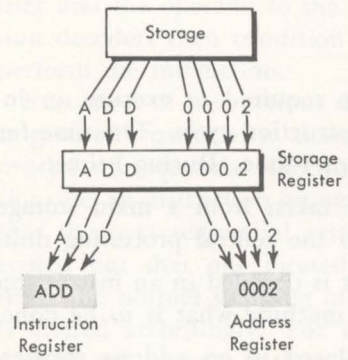


Figure 62. Register Nomenclature and Function

Registers differ in size, capacity, and use. In some cases, extra positions detect possible overflow conditions during an arithmetic operation. For example, if two 11-digit numbers are added, it is possible that the result is a 12-digit answer (Figure 63). In this figure, register A holds one factor and register B holds the other factor. The two factors are combined and the result is placed in register C, where an overflow condition is indicated by the presence of data in the overflow position. The contents of other registers can be shifted right or left within the register and, in some cases, even between registers. Figure 64 shows shifting of register contents three positions to the right. Positions vacated are filled with zeros and numbers shifted beyond register capacity are lost. Numbers can also be shifted out of one end of the register and into the opposite end, as shown.

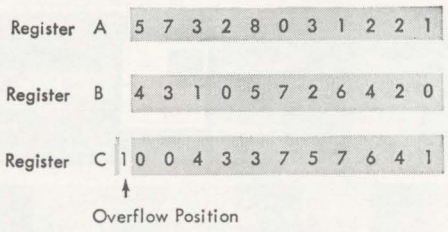


Figure 63. Overflow Condition Resulting from Addition

In other instances, a register holds data while associated circuits analyze the data. For example, an instruction can be placed in a register, and associated circuits can determine the operation to be performed and locate the data to be used. Data within specific registers may also be checked for validity.

The more important registers of a system, particularly those involved in normal data flow and storage addressing, have small incandescent or neon lights associated with them. These lights are located on an operator console (Figure 65) for visual indication of register contents and various program conditions.

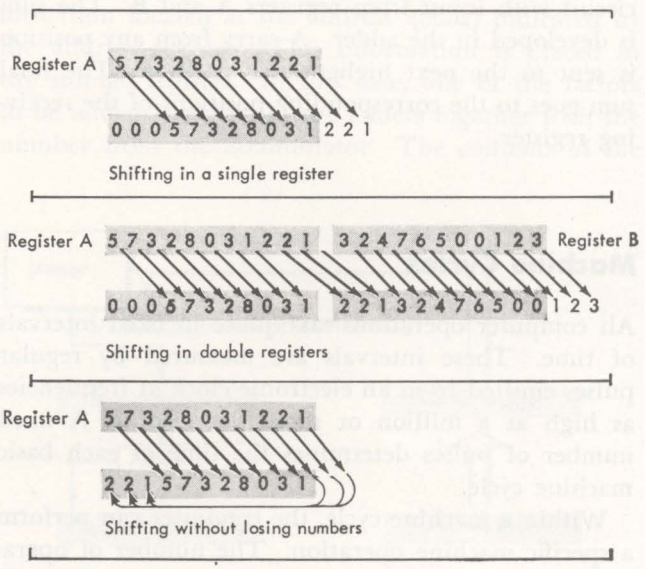


Figure 64. Types of Computer Register Shifting

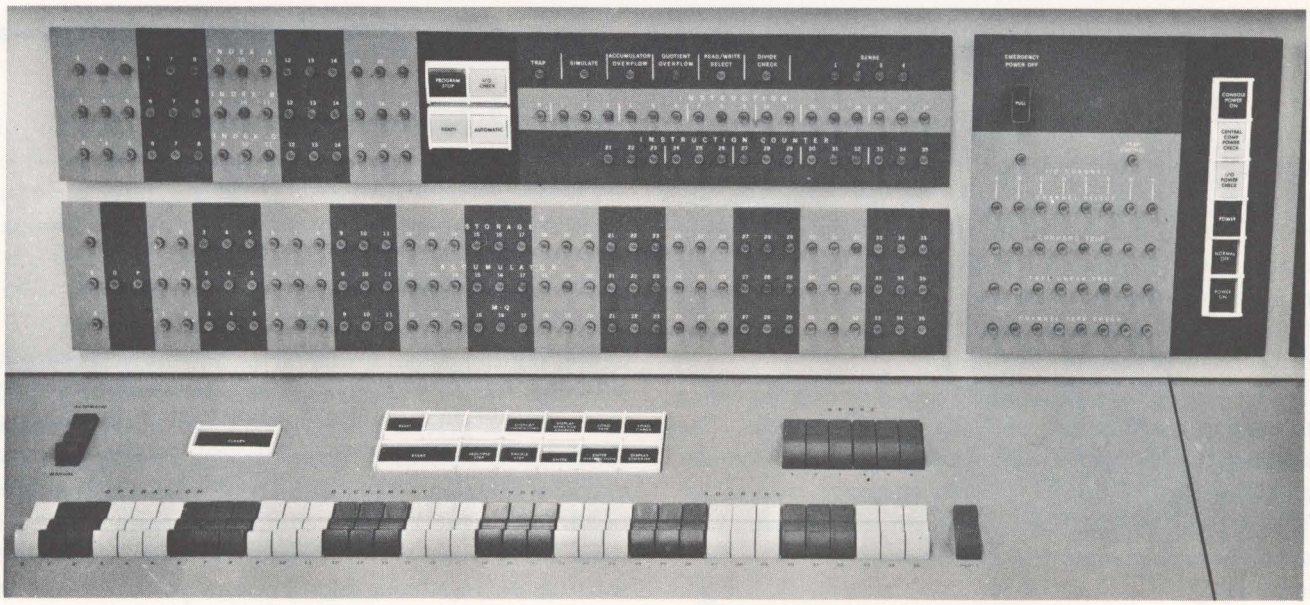


Figure 65. Operator Console Indicators

Counter

The counter is closely related to a register, and usually performs the same functions. In addition, its contents can be increased or decreased by some amount. The action of a counter is related to its design and use within the computer system. Like the register, it may also have visual indicators on the operator console.

Adder

The adder receives data from two or more sources, performs addition, and sends the result to a receiving register. Figure 66 shows two positions of an adder circuit with input from registers A and B. The sum is developed in the adder. A carry from any position is sent to the next higher-order position. The final sum goes to the corresponding positions of the receiving register.

Machine Cycles

All computer operations take place in fixed intervals of time. These intervals are measured by regular pulses emitted from an electronic clock at frequencies as high as a million or more per second. A fixed number of pulses determines the time of each basic machine cycle.

Within a machine cycle, the computer can perform a specific machine operation. The number of operations required to execute a single instruction depends on the instruction. Various machine operations are thus combined to execute each instruction.

Instructions consist of at least two parts, an operation and an operand. The operation tells the machine which function to perform: read, write, add, subtract, and so on. The operand can be the address of either data or an instruction or the address of an input output unit or other device. It can also specify a control function, such as shifting a quantity in a register, or backspacing and rewinding a reel of tape.

To receive, interpret, and execute instructions, the central processing unit must operate in a prescribed sequence. The sequence is determined by the specific instruction and is carried out during a fixed interval of timed pulses.

Instruction Cycle

The first machine cycle required to execute an instruction is called an instruction cycle. The time for this cycle is instruction or I-time. During I-time:

1. The instruction is taken from a main storage location and brought to the central processing unit.

2. The operation part is decoded in an instruction register. This tells the machine what is to be done.

3. The operand is placed in an address register. This tells the machine what it is to work with.

4. The location of the next instruction to be executed is determined.

At the beginning of a program, an instruction counter is set to the address of the first program instruction. This instruction is brought from storage

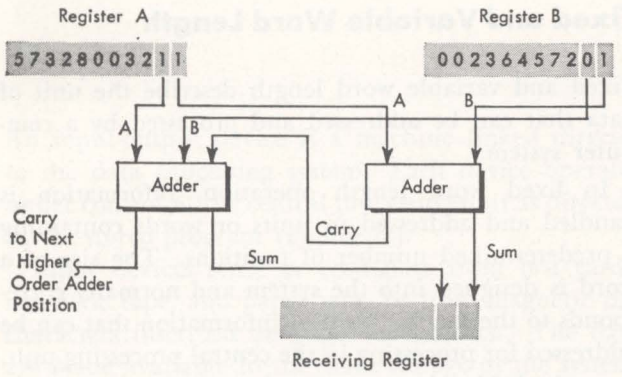


Figure 66. Adders in a Computer System

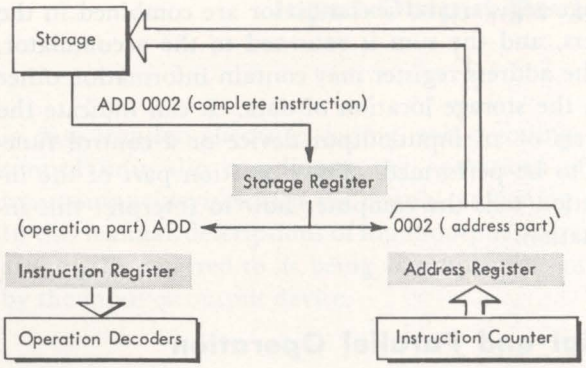


Figure 67. Computer I Cycle Flow Lines

and, while it is being executed, the instruction counter automatically advances (steps) to the location corresponding to the space occupied by the next stored instruction. If each instruction occupies one word position, the counter steps one; if an instruction occupies five storage positions, the counter steps five. By the time one instruction is executed, the counter has located the next instruction in program sequence. The stepping action of the counter is automatic. In other words, when the computer is directed to a series of instructions, it will execute these one after another until instructed to do otherwise.

Assume that an instruction is given to add the contents of storage location 2 to the contents of the accumulator register. Figure 67 shows the main registers involved and the information flow lines.

I-time begins when the instruction counter transfers the location of the instruction to the address register. This instruction is selected from storage and placed in a storage register. From the storage register, the operation part is routed to the instruction register and the operand to the address register. Operation decoders then condition proper circuit paths to perform the instruction.

Execution of instructions does not necessarily have to proceed sequentially. Certain instructions alter the process of sequential execution unconditionally. In this case, an instruction brought from storage indicates that the next sequential instruction is not to be executed but that one located in another position is next; the normal stepping of the instruction counter is altered accordingly. For instance, the instruction counter can be reset back to the beginning of the program so that the entire program can be repeated for another incoming group of data.

This branching (transfer) to alternative instructions may also be conditional. The computer can be directed to examine some indicating device and then branch if the indicator is on or off. Such an instruction says, in effect, "Look at the sign of the quantity in the accumulator; if this sign is minus, take the next instruction from location 5000; if the sign is plus, proceed to the next instruction in sequence." The instruction counter is set according to one of the *two* possible storage locations (5000, or the location of the next instruction in sequence). The logical path followed by the computer (that is, the precise sequence of instructions executed) may be controlled either by unconditional branching or by a series of conditional tests applied at various points. However, the arrangement of instructions in storage is not normally altered.

Execution Cycle

I-time is usually followed by one or more machine cycles which occur during execution or E-time. The number of execution cycles required depends on the instruction to be executed. Figure 68 shows the data flow following I-time illustrated by Figure 67.

The E-cycle starts by removing from storage the information located at the address (0002) indicated by the address register. This information is placed in the storage register. In this case, one of the factors to be added is placed in the adders together with the number from the accumulator. The contents of the

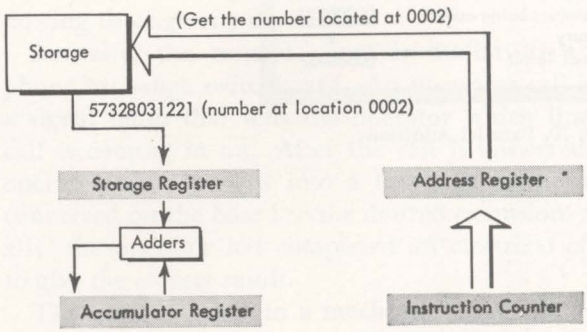


Figure 68. Computer E Cycle Following an I Cycle

storage register and accumulator are combined in the adders, and the sum is returned to the accumulator.

The address register may contain information other than the storage location of data. It can indicate the address of an input-output device or a control function to be performed. The operation part of the instruction tells the computer how to interpret this information.

Serial and Parallel Operation

Computers are classified as either serial or parallel depending on the method the computer uses to perform arithmetic. Essentially, all arithmetic is performed by addition.

In a serial computer, numbers to be added are considered one position at a time (the units position, tens position, hundreds, and so on) in the same way that addition is done with paper and pencil. Whenever a carry is developed, it is retained temporarily and then added to the sum of the next higher-order position.

The time required for serial operation depends on the number of digits in the factors to be added. Serial addition is shown in Figure 69.

In a parallel computer, addition is performed on complete data words. The words are combined in one operation, including carries. Any two data words, regardless of the magnitude of the numbers contained in the words, can be added in the same time. Figure 70 shows parallel addition.

	1st Step	2nd Step	3rd Step	4th Step
Addend	1234	1234	1234	1234
Augend	2459	2459	2459	2459
Carry	1	1		
Sum	3	93	693	3693

Figure 69. Serial Addition

	00564213
Numbers being added	00000824
Carry	1
Final Result	00565037

Figure 70. Parallel Addition

Fixed and Variable Word Length

Fixed and variable word length describe the unit of data that can be addressed and processed by a computer system.

In fixed word length operation, information is handled and addressed in units or words containing a predetermined number of positions. The size of a word is designed into the system and normally corresponds to the smallest unit of information that can be addressed for processing in the central processing unit. Records, fields, characters, or factors are all manipulated in parallel as words, and registers, counters, accumulators, and storage are designed to accommodate a standard word.

In variable word length operations, data handling circuitry is designed to process information serially as single characters. Records, fields or factors may be of any practical length within the capacity of the storage unit. Information is available by character instead of by word.

Operation within a given data processing system may be entirely of a fixed word nature, entirely variable, or a combination of both.

In the IBM 7090 Data Processing System, data are stored and processed as 36-bit words; all data manipulation operations, including arithmetic, are done in parallel. However, provision is made to select, to shift, and to perform logical operations on portions of words. Consequently, the size of the unit of data within a word can be adjusted.

In the IBM 705 or 7080 Data Processing System, data are stored and processed as single characters. All arithmetic and the majority of data-handling operations are done serially, character by character, but some operations such as moving records from one area of storage to another are done in parallel as fixed words of five characters. Instructions are also stored and interpreted as five-character words.

In the IBM 7070 Data Processing System, data are stored and processed as fixed words of 10 numerical digits with sign. All arithmetic operations are performed in serial fashion, while most data handling is done in parallel by words. However, there is also provision for making individual digits and portions of words accessible.

An input-output device is a machine linked directly to the data processing system. Each device operates under control of the central processing unit as directed by the stored program (Figure 71).

Input devices sense or read data from IBM cards, magnetic tape, and paper tape, or from magnetic ink characters inscribed on paper documents. The data are made available to the main storage of the system. Output devices record or write information from main storage on IBM cards, magnetic tape, and paper tape, or prepare printed copy. Some systems also display output on a cathode ray tube.

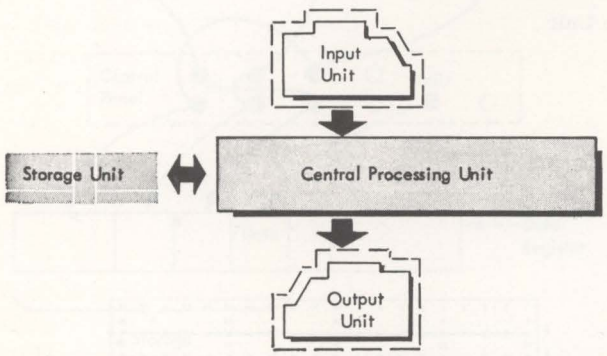


Figure 71. Input-Output Units in the Data Processing System

Reading and Writing

Reading takes place as the input medium physically moves through an input device. Information is sensed or read and is converted to a form compatible with the computer system. The information is then transmitted to main storage.

Writing involves converting data from primary storage to a form or language compatible with an output medium and recording the data using an output device.

Most input-output devices are automatic; once started, they continue to operate as directed by the stored program. Instructions in the program select the required device, direct it to read or write, and indicate the storage location that data will be put into or taken from.

A few input devices are manual, and no medium for recording data is involved. Instead, data are entered directly into storage using a keyboard or switches that are usually a part of the operator console.

In some data processing systems, certain input-output devices are connected to the system through a control unit that contains many of the circuits involved in data transfer, checking, coding, and decoding. The control unit also coordinates the operations of the input-output devices with the central processing unit. In this manual, descriptions of input-output operations are usually referred to as being accomplished entirely by the input or output device.

Validity Checks

All data transferred between the input-output units and storage are automatically checked for validity in two ways. First, the data are checked before being sent by the input device and are also checked when received by the output device. Second, certain data checks are also made within the central processing unit as it receives or sends data. These checks do not detect the use of wrong data. For example, if a 5 is entered instead of a 4, this error cannot be detected by the machine. However, if the indicated number or character is represented or coded incorrectly on the medium or within the machine, this is automatically detected.

Indicators, Keys, and Switches

All input-output units have indicator lights and operating keys and switches (Figure 72). The indicator lights show the status of a unit: on, off, ready, selected, and so on. The operating keys and switches are used primarily to start and stop operations manually. The specific functions and use of the indicators, keys, and switches are described in the IBM manuals for particular machines and systems.

Control Panel

Many input-output devices are equipped with a control panel (Figure 73). The panel provides a means of editing, rearranging, deleting, and selecting data flowing through the device.

Basically, the control panel is similar to a telephone exchange switchboard. An incoming call lights a signal lamp that tells the operator which line the call is coming in on. After the call is answered, the operator plugs a cord into a hub that is internally connected on the board to the desired extension. Actually, the operator has completed an electrical circuit to give the correct result.

The control panel in a machine completes circuits internally by means of wires placed in the panel. The actual connections in the panel are made through

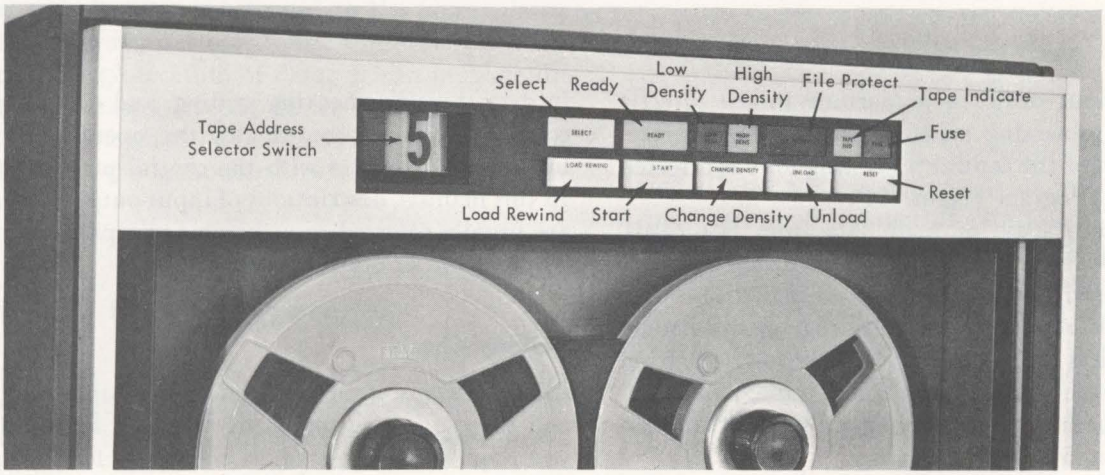
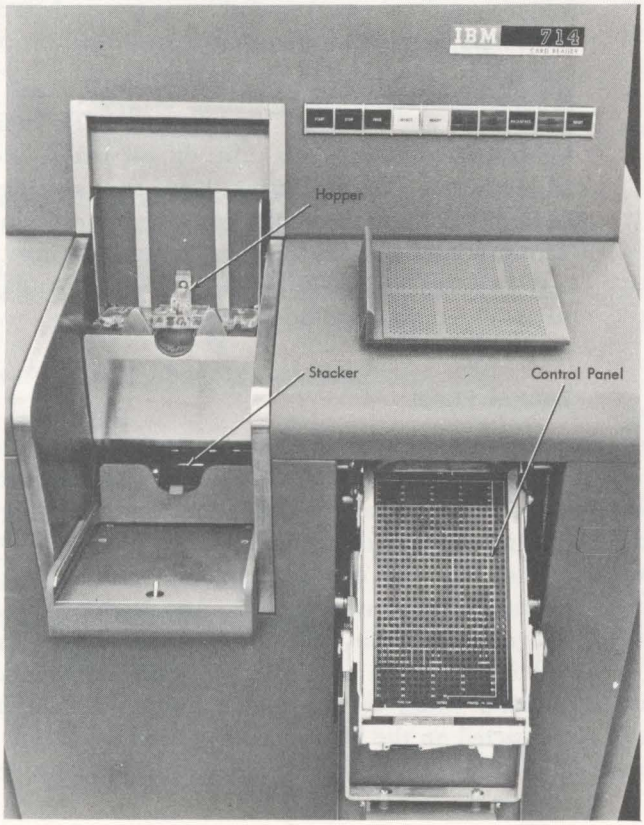


Figure 72. Operating Keys and Lights, IBM 729 II Magnetic Tape Unit



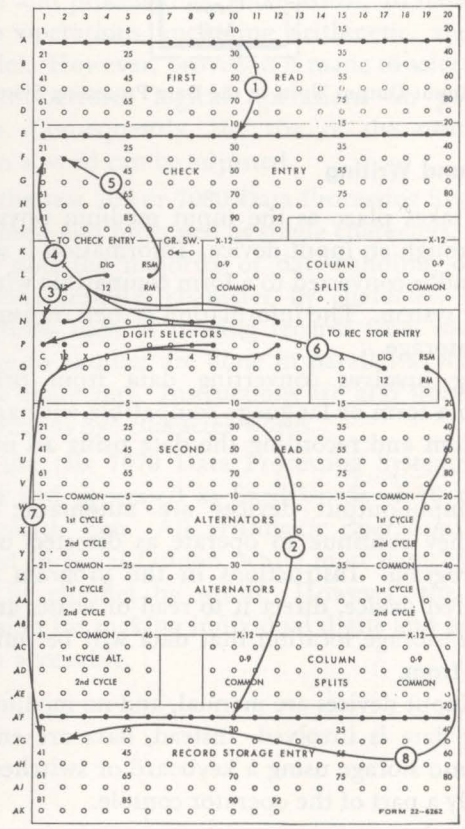


Figure 73. IBM 714 Card Reader with Control Panel Schematic

holes called exit and entry hubs (Figures 74 and 75). An exit hub is one that emits an impulse, an entry hub is one that accepts an impulse. The exit and entry hubs used depend on the functions to be accomplished. The panel is prewired by the operator for a specific job before it is placed in the machine.

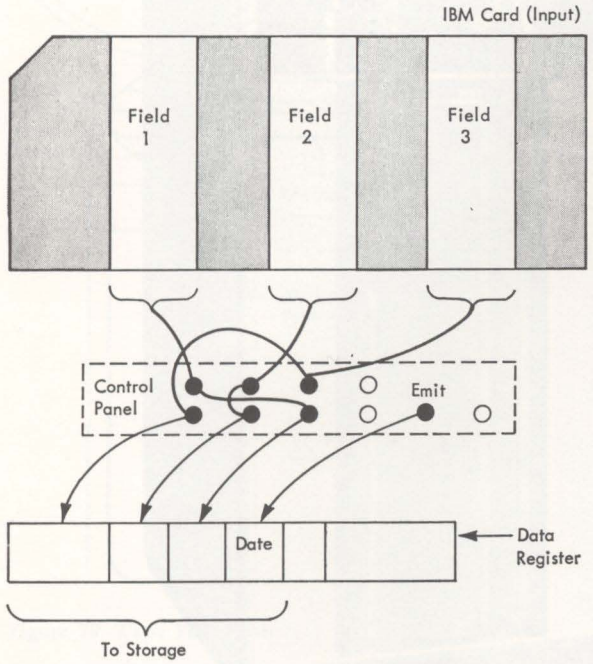


Figure 74. Card Reader Control Panel Function

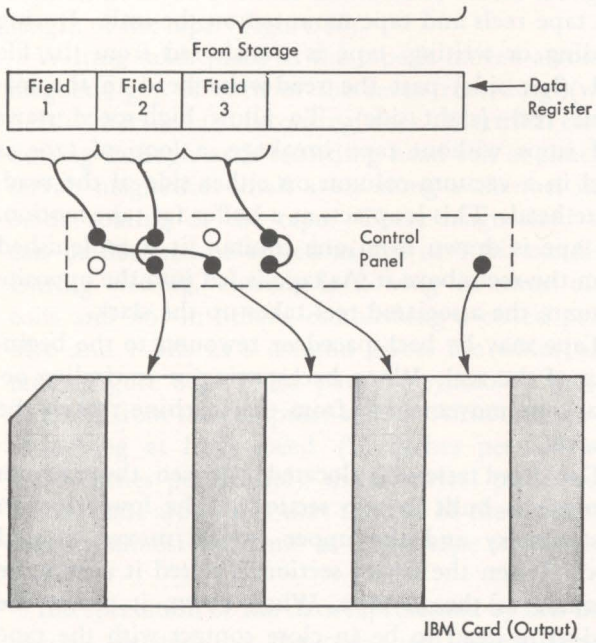


Figure 75. Card Punch Control Panel Function

Input units with control panels can alter or change data after the data are read by the unit but before the data are sent to the main storage of the system (Figure 74). Output units alter or change data after the data are received from storage but before the data are punched or printed (Figure 75).

Control panels are removable and can be readily changed for different procedures, or a separate panel may be used for each operation.

Card Readers

Card reading devices introduce IBM punched card data into the computer system. The card reader moves or feeds cards past a reading unit that converts the data on the card into an electronic form. Two types of reading units are used: reading brushes or photoelectric cells.

In the brush type reader, cards are mechanically moved from a card hopper, through the card feed unit, and under reading brushes. The reading brushes electrically sense the presence or absence of holes in each column of the card (Figure 76). This electric sensing converts the information of the card to electrical impulses that can be utilized by the card reader circuitry and stored as data. After the cards are read they are moved from the card feed unit and placed in the card stacker in the same sequence in which they were fed into the reader. Some card readers have two sets of reading brushes; each card can be read twice as it moves through the card feed unit as a check on the validity of the reading process.

The photoelectric type of card reader performs the same functions as the brush type; the difference is in

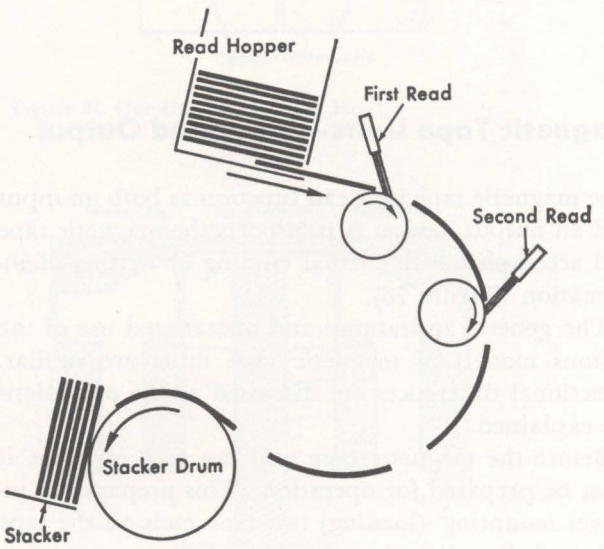


Figure 76. Read Feed

the method of sensing the holes. Photoelectric cells are activated by the presence of light. As the punched card is passed over a light source in the card reader, light passing through the punched holes activates photoelectric cells, one cell for each column of the card.

Card reading speed varies from 100 to 1000 cards a minute, depending on the type of card reader.

Card Punches

Output from the computer system is recorded in IBM cards by a card punching device. The card punch automatically moves blank cards, one at a time, from the card hopper, under a punching mechanism that punches data received from storage (Figure 77). After the card is punched, it is moved to a checking station where the data are read and checked with the information received at the punching station. The card is then moved to the stacker.

Card punching speed varies from 100 to 250 cards per minute depending on the type of card punch.

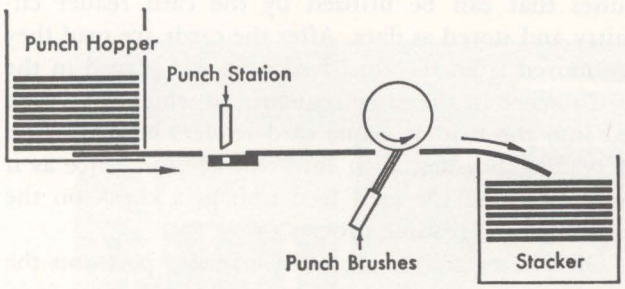


Figure 77. Punch Feed

Magnetic Tape Units — Input and Output

The magnetic tape unit can function as both an input and an output device; it transports the magnetic tape and accomplishes the actual reading or writing of information (Figure 78).

The general appearance and operational use of the various models of magnetic tape units are similar. Functional differences are discussed as the operations are explained.

Before the magnetic tape unit can read or write, it must be prepared for operation. This preparation involves mounting (loading) two tape reels on the tape unit and threading the tape through the tape moving (feed) mechanism. Figure 79 shows the location of

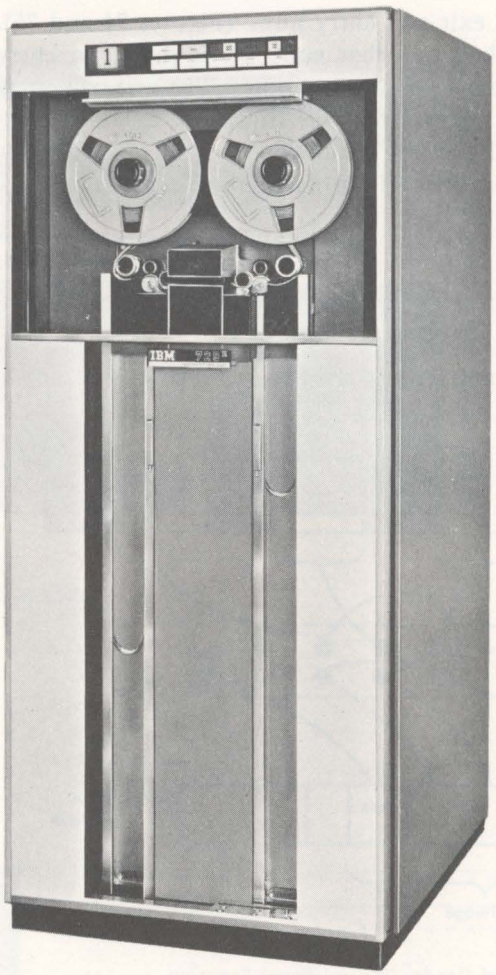


Figure 78. IBM 729 IV Magnetic Tape Unit

the tape reels and tape mounted on the unit. During reading or writing, tape is transferred from the file reel (left side) past the read-write head to the machine reel (right side). To allow high-speed starts and stops without tape breakage, a loop of tape is held in a vacuum column on either side of the read-write head. This loop acts as a buffer for tape motion. As tape is drawn from one column, it is replenished from the reel above it. As tape is fed into the opposite column, the associated reel takes up the slack.

Tape may be backspaced or rewound to the beginning of the reel. When backspacing or rewinding occurs, tape movement is from the machine reel to the file reel.

The head assembly located between the vacuum columns is built in two sections. The lower section is stationary and the upper section moves up and down. When the upper section is raised it allows the operator to thread tape. When down, it causes the read-write head to be in close contact with the tape for reading and writing.

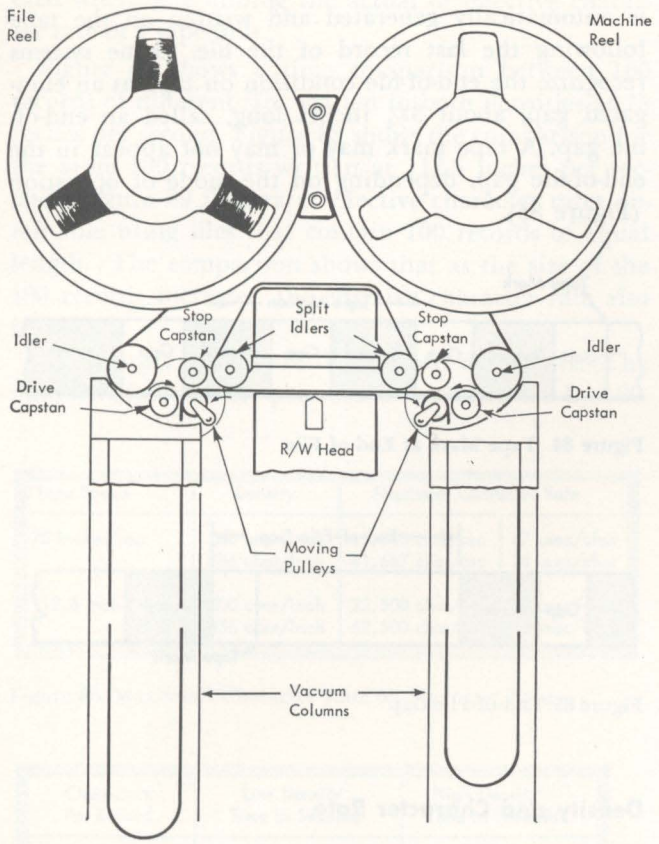


Figure 79. Tape Feed Unit

Writing and Reading Magnetic Tape

Data are recorded on tape as magnetized spots or bits located in seven parallel channels or tracks along the length of the tape (Figure 80).

Writing takes place as the tape is moved across the magnetic gap of each of the seven recording (write) heads—one for each recording track. Electrical current flowing through each recording head coil at timed intervals magnetizes small areas along a channel in the oxide coating of the tape. These magnetized areas can be detected or sensed as a 0 or 1 bit condition. During writing, current is flowing in some of the coils and not in others, establishing a coded pattern of 0 and 1 bits in a column across the width of the tape (Figure 80). These patterns symbolize the data received from the computer system. Although the tape is moving at high speed (75 inches per second or 112.5 inches per second), the electrical pulses to the write heads are so fast that the size of the magnetized areas is almost the same as if the tape were motionless.

Interpretation of the data represented by bits in one column depends on the code used in writing. For example, the bits of one column of tape written in the seven-bit alphameric code represent one character; the bits of one column written in the binary system represent part of a binary word. The spacing between columns of bits is automatically determined by the magnetic tape unit used in writing.

Two types of reading and writing heads are used in present magnetic tape units. One type, the one-gap head, has only one magnetic gap for each of the seven heads. Both reading and writing occur at this one gap (Figure 81). The second type is the newer two-gap head which has two magnetic gaps for each of the seven heads (Figure 82). Writing occurs at one gap and reading occurs at the other. The two-gap head

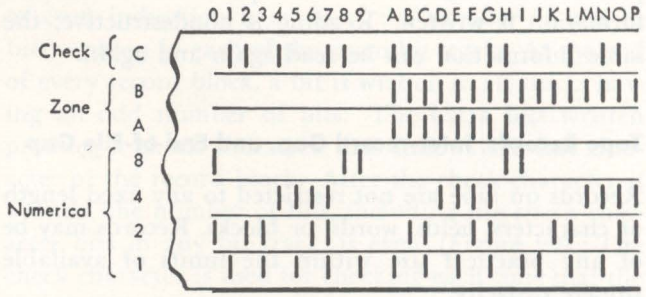


Figure 80. Magnetic Spots on Tape

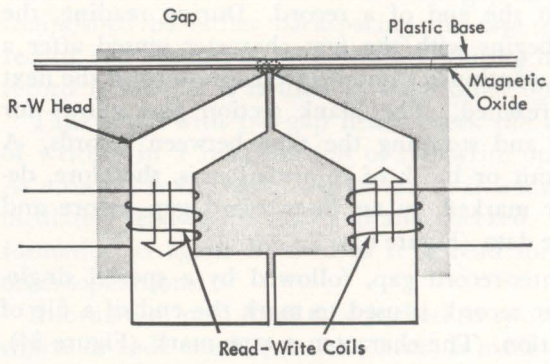


Figure 81. One-Gap Read-Write Head

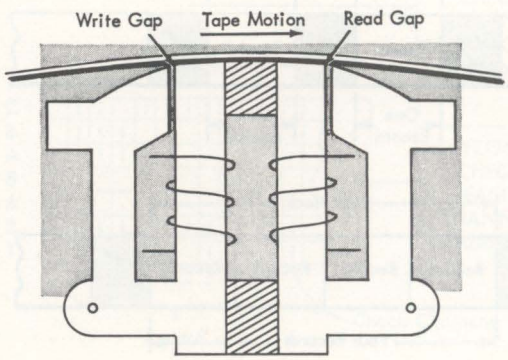


Figure 82. Two-Gap Read-Write Head

offers advantages that are discussed in the tape validity checking sections. However, the general principle of writing and reading are the same, regardless of the head used.

Reading information from tape is accomplished as the tape is moved past the read gap of the two-gap head or the read-write gap of the one-gap head. As the magnetized areas pass the gap, small currents are generated in the read coil of the head. These electrical currents or pulses from the magnetized areas on tape symbolize data in electronic form and are made available by tape unit circuitry to the computer system.

Writing on magnetic tape is destructive; that is, previous information on the tape is lost as new information is written. Reading is nondestructive; the same information can be read again and again.

Tape Records, Inter-record Gap, and End-of-File Gap

Records on tape are not restricted to any fixed length of characters, fields, words, or blocks. Records may be of any practical size within the limits of available storage capacity.

Records or groups of data are separated on tape by a record gap—a length of blank tape about ¾ inch long. During writing, the gap is automatically produced at the end of a record. During reading, the record begins with the first character sensed after a gap and continues without interruption until the next gap is reached. The blank section also allows for starting and stopping the tape between records. A single unit or block of information is, therefore, defined or marked by an inter-record gap before and after the data (Figure 83).

An inter-record gap, followed by a special single-character record, is used to mark the end of a file of information. The character, a tape mark (Figure 84),

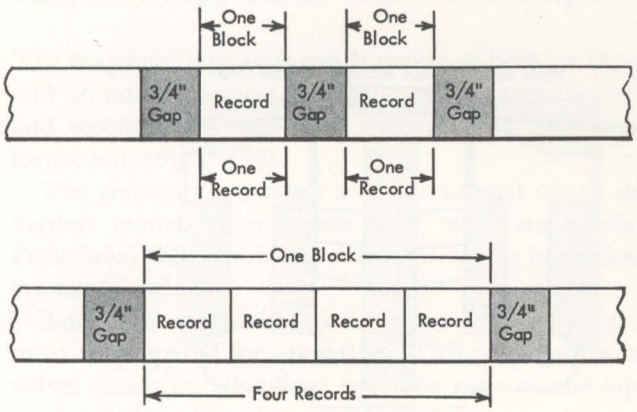


Figure 83. Single and Multiple Record Blocks

is automatically generated and written on the tape following the last record of the file. Some systems recognize the end-of-file condition on tape as an elongated gap, about 3¾ inches long, called an end-of-file gap. A tape mark may or may not appear in the end-of-file gap, depending on the mode of operation (Figure 85).

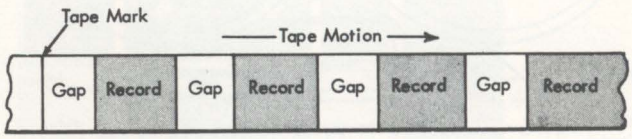


Figure 84. Tape Mark at End of File

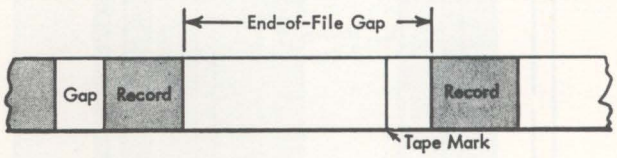


Figure 85. End-of-File Gap

Density and Character Rate

The major differences in IBM magnetic tape units are the speed at which tape is moved through the tape unit and the density of the recorded information on tape.

The speed of a tape unit is stated as the length of tape that is transported or moved over the read-write head in a unit of time. Two speeds are currently used: 75 inches per second and 112.5 inches per second.

Density is the greatest possible number of bits in a track or the number of columns of data for a unit length of tape. The present densities used are 200 or 556 columns of bits per inch of tape.

The faster the tape speed and the greater the density of recording, the higher becomes the rate at which information is recorded on or read from tape. Information rate for a tape unit constitutes the number of columns of bits of data read or written in a unit of time. By combining speed and density, a maximum character or word rate per second is determined (Figure 86).

Because a record gap is placed between each record or block of records on tape, the total time required to read a record must include time to space over the inter-record gap. The time required to space over the gap is called access time to the data. Average access time for tape moving at 75 inches a second is 10.8 milliseconds and for tape moving at 112.5 inches per second, 7.3 milliseconds. Access time must be consid-

ered when determining the actual or effective character rate of a tape unit.

Figure 87 shows a time comparison between 100 records of different size written in both densities at 75 inches per second. Figure 88 shows the comparison for the same 100 records written at 112.5 inches per second. Figure 89 illustrates effective character rates obtainable using files that contain 100 records of equal length. The comparison shows that as the size of the 100 records increases, the effective character rate also increases.

The results shown are determined as follows: The number of characters per record is multiplied by 100.

Tape Speed	Density	Maximum Character Rate	
75 inches/sec	200 char/inch	15,000 char/sec	67 usec/char
	556 char/inch	41,667 char/sec	24 usec/char
112.5 inches/sec	200 char/inch	22,500 char/sec	44 usec/char
	556 char/inch	62,500 char/sec	16 usec/char

Figure 86. Maximum Character Rate on Magnetic Tape

Characters Per Record	Low Density Time in Seconds	High Density Time in Seconds
360	3.49	1.94
720	5.90	2.81
1800	13.14	5.40
3600	25.20	9.72

Figure 87. Time Comparison for 75 Inches per Second Tape

Characters Per Record	Low Density Time in Seconds	High Density Time in Seconds
360	2.31	1.30
720	3.90	1.88
1800	8.65	3.61
3600	16.57	6.49

Figure 88. Time Comparison for 112.5 Inches per Second Tape

Characters Per Record	Low Density Characters/second	High Density Characters/second
Tape Speed (75"/second)		
360	10315	18556
720	12203	25623
1800	13699	33333
3600	14286	37037
Tape Speed (112.5"/second)		
360	15584	27692
720	18461	38298
1800	20809	49861
3600	21726	55470

Figure 89. Effective Character Rates for 100 Records

This product is divided by the appropriate amount from Figure 87 or Figure 88. The result is an effective character rate. For example: Find the effective character rate for 100 records of 360 characters each recorded at 75 inches per second and at a density of 200 characters per inch. The number of characters per record (360) is multiplied by 100 (36,000). The time in seconds for one hundred 360-character records, (3.49) is taken from Figure 87. Dividing 36,000 by 3.49 gives a result of 10,315 characters per second.

Checking BCD Tape

Information written on tape is automatically checked for validity. As a record block is being written, an odd or even indication is made of the number of bits (1 bits) written in each of the seven bit tracks. At the end of every record block, a bit is written in all tracks having an odd number of bits. The extra bits written produce a check character that follows the last character of the record block. After the check character is written, the number of bits, including the check character bits in any bit track, is even (Figure 90). The check character is used for checking each time that the record is read.

Checks on the validity of data are accomplished when the information is read from tape. The validity checking of data recorded with a one-gap head is accomplished by either backspacing the tape over the record and reading it or by simply waiting until the record is read as it is used in some other operation.

Tape units with two-gap heads check the accuracy of writing in a read portion of the write operation. As a record is written at the write gap it is almost immediately read at the read gap and checked. The information is again checked as it is read for use in other operations.

Information read from tape is checked two ways. A character code check (vertical check) is made on each column of information to insure that an even num-

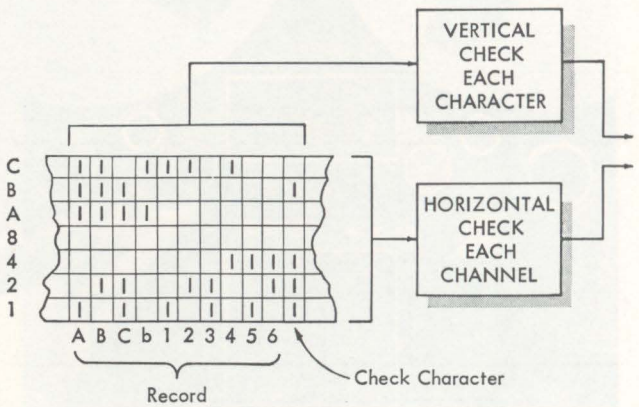


Figure 90. Magnetic Tape Check Character

ber of bits exists for each character read. If an odd number of bits is detected for any character or column of bits, an error is indicated. A longitudinal record check is made by developing an odd or even indication of the number of bits read in each of the seven bit tracks of the record, including the bits of the check character. If any bit track of the record block indicates an odd number of bits after it is read, an error is indicated.

The two-gap head provides an additional method of checking called dual-level sensing (Figure 91). A critical analysis is made of the signal strength of the recorded information. On the basis of this analysis, recorded information is accepted if it meets certain standards. If it does not, corrective actions are taken to improve it. If the corrective action fails to sufficiently correct the signal level, an error is indicated.

Checking Binary Tape

A tape written in binary form is checked for accuracy of writing using the same methods as those used with BCD tapes. The difference is that each column of bits across the tape and the check character must have an odd number of bits (vertical check). Each bit track of a written record, however, should have an even number of bits, including any bits of the check character (horizontal check). If these conditions are not satisfied, an error is indicated.

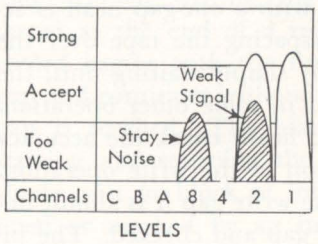


Figure 91. Signal Strength

Photosensing Markers

Photosensing markers called reflective strips are placed on the tape by the operator to enable the tape unit to sense the beginning and the end of the usable portion of tape. Photoelectric cells in the tape unit sense the markers as either the load point marker (where reading or writing is to begin) or as the end-of-reel marker (where writing is to stop).

The markers (Figures 92 and 93) are small pieces of plastic, 1 inch by $\frac{3}{16}$ inch, coated with vaporized aluminum on one side and with adhesive on the other. They are fastened to the base (uncoated) side of the tape.

Load Point Marker

At least ten feet of tape must be allowed between the beginning of the reel and the load point marker as a leader for threading the tape on the tape unit. More than ten feet may be allowed by placing the marker at any desired distance from the beginning of the reel. To indicate the load point, the 1-inch dimension of the marker must be parallel to, and not more than $\frac{1}{32}$ inch from, the channel 1 edge of the tape (the edge nearest the operator when the reel is mounted). See Figure 92.

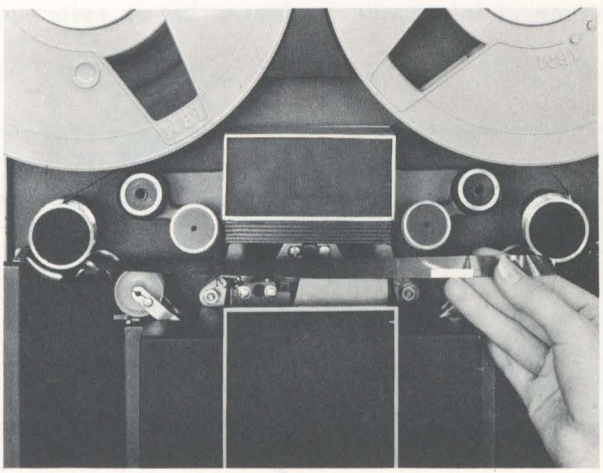


Figure 92. Load Point Marker

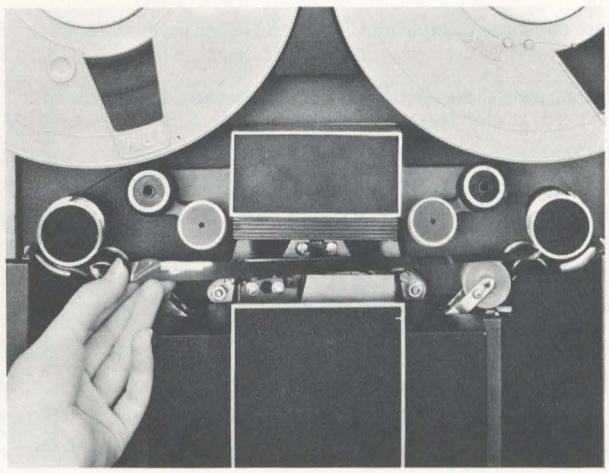


Figure 93. End-of-Reel Marker

End-of-Reel Marker

About 18 feet of tape are usually reserved between the end-of-reel marker and the end of the tape. This space includes at least ten feet of leader and enough tape to hold a record after the end-of-reel marker is sensed. Any usable length over the ten feet may be allowed to permit additional records to be written after the marker is sensed. To indicate end of reel, the marker must be placed parallel to, and no more than $\frac{1}{32}$ inch from, the C track edge of the tape (the edge nearest the tape unit when the reel is mounted). See Figure 93.

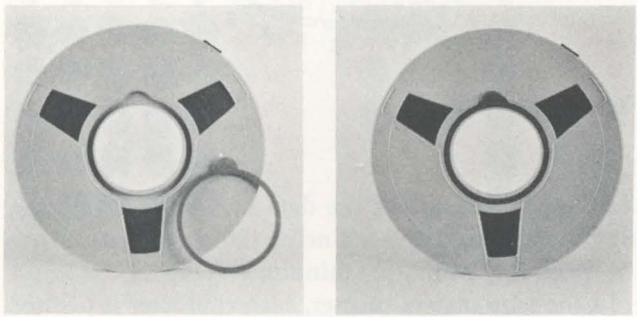


Figure 94. File Protection Device

File Protection

Because writing automatically destroys any previous information on the tape, a file protection device is used to prevent accidental erasure of information. This device is used when tapes are to be saved for further reference.

The file protection device is a plastic ring that fits into a round groove molded in the tape reel (Figure 94). When the ring is in place, either reading or writing can occur. When the ring is removed, writing is suppressed and only reading can take place; thus, the file is protected from accidental erasure.

Paper Tape Reader

The paper tape reader reads data represented as punched holes in a paper tape and, when used as an input device, transmits the data to main storage. The tape reader (Figure 95) moves or feeds the tape past a reading unit. The presence or absence of holes in the tape is sensed and converted to electronic impulses that are used as data by the computer system. Accuracy of reading is determined by making a parity check on each character used (8 channel code only). The speed of reading, 150 or 500 characters per second, depends on the type of paper tape reader.

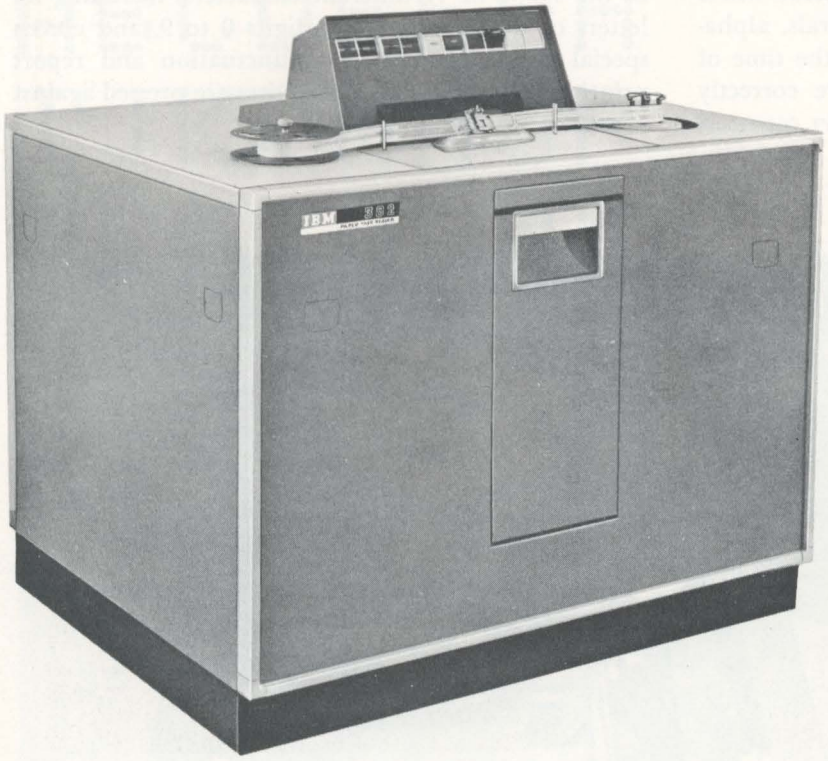


Figure 95. IBM 382 Paper Tape Reader

Paper Tape Punch

Data from the computer system are recorded as punched holes in paper tape by an automatic tape punch (Figure 96). Data received from main storage are converted to a tape code and punched in blank tape as the tape is moved through a punching mechanism. Accuracy of data recorded is verified by a parity check for each character (8 channel code only). Tape is punched at a density of ten characters to the inch and at a rate of 15 characters per second.

Printers

IBM printing devices provide a permanent visual record of data from the computer system. Speeds of printing vary from 10 to 2,000 characters per second.

As an output unit, the printer receives data, symbolized in electronic form, from the computer system. These electronic symbols enter appropriate circuitry and cause printing elements to be actuated. All printing devices have a paper transport that automatically moves the paper as printing progresses.

The major printing devices consist of the print wheel printer, wire matrix printer, chain printer, and the typewriter.

The print wheel printer is equipped with 120 rotary print wheels (Figure 97). Each print wheel has 48 characters of type including numerals, alphabetic symbols, and special characters. At the time of printing, all of the 120 print wheels are correctly positioned to represent the data to be printed. Printing occurs as a complete line of 120 characters. Printing speed is 150 lines per minute.

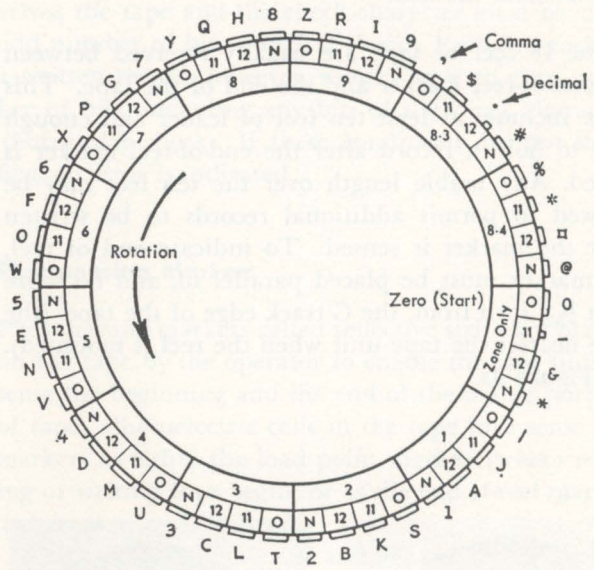


Figure 97. Print Wheel

In the wire matrix printer, each character is printed as a pattern of dots formed by the ends of small wires arranged in a five-by-seven rectangle (Figure 98). By extending selected wires, the patterns may be arranged in the shape of 47 different characters, including all letters of the alphabet, the digits 0 to 9, and eleven special characters used for punctuation and report printing (Figure 99). Selected wires are pressed against

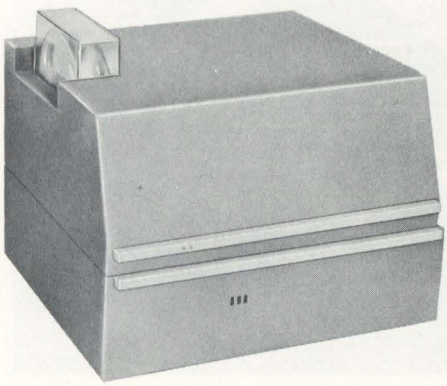
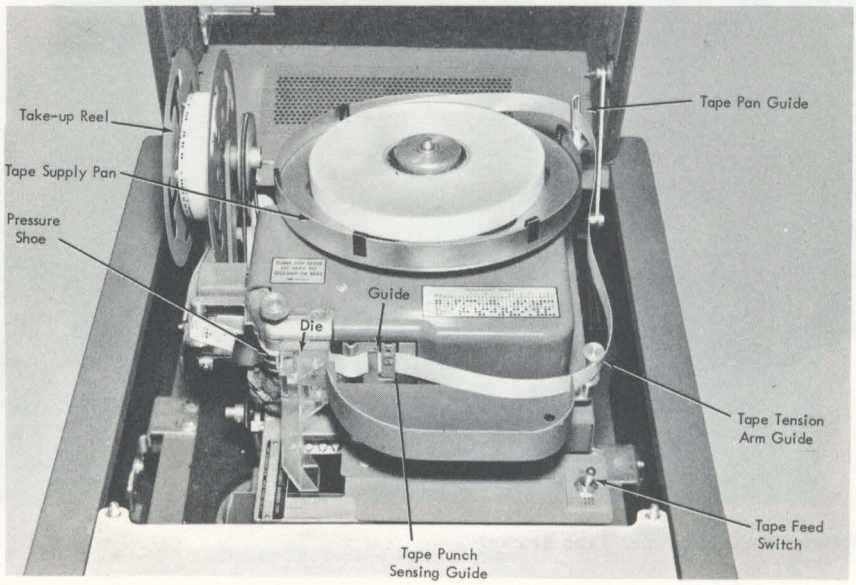


Figure 96. IBM 962 Tape Punch



an inked fabric ribbon to print the characters on paper. Characters are printed 120 to the line and at a rate of 500 or 1000 lines per minute, depending upon the model of printer.

The chain printer is an electromechanical line printer using engraved type. Alphabetic, numeric, and special characters are assembled in a chain (Figure 100). As the chain travels horizontally, each character is printed as it is positioned opposite a magnetically actuated hammer that presses the paper against one piece of type in the moving chain. Up to 132 positions may be printed on one line at a speed of 600 lines per minute. The print chain can be easily changed to provide a choice of print fonts.

The typewriter used as an output device (Figure 101) is similar to the ones used manually. The major difference is that control of the typewriter and the printing are accomplished automatically as directed by the stored program. Printing speed is about 600 characters per minute; spacing and carriage return are automatic.

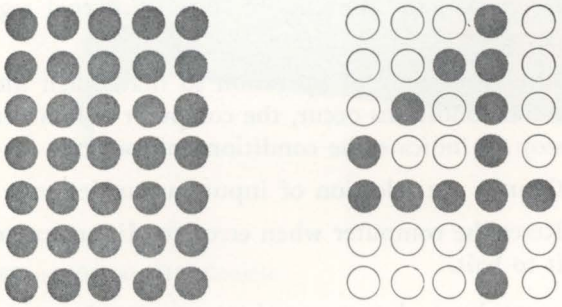


Figure 98. 5 x 7 Dot Pattern

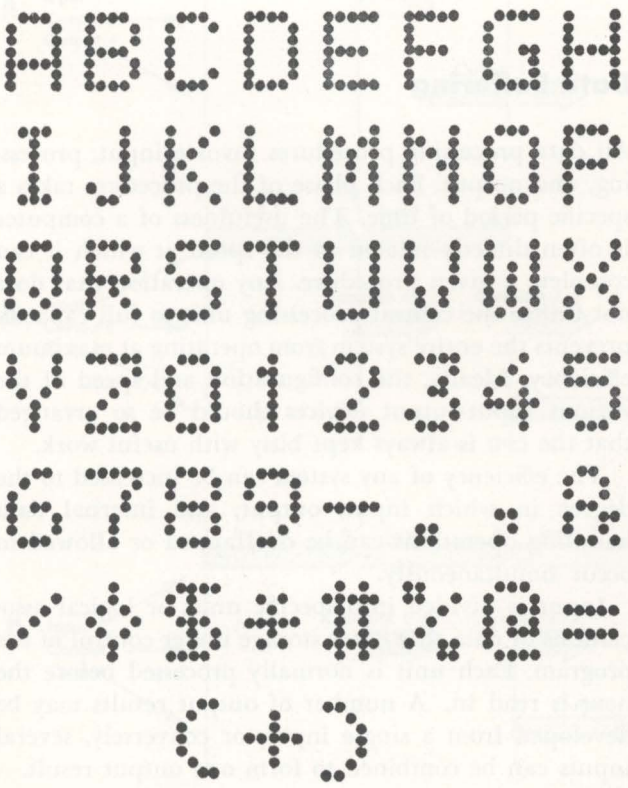


Figure 99. Wire Printing Dot Patterns

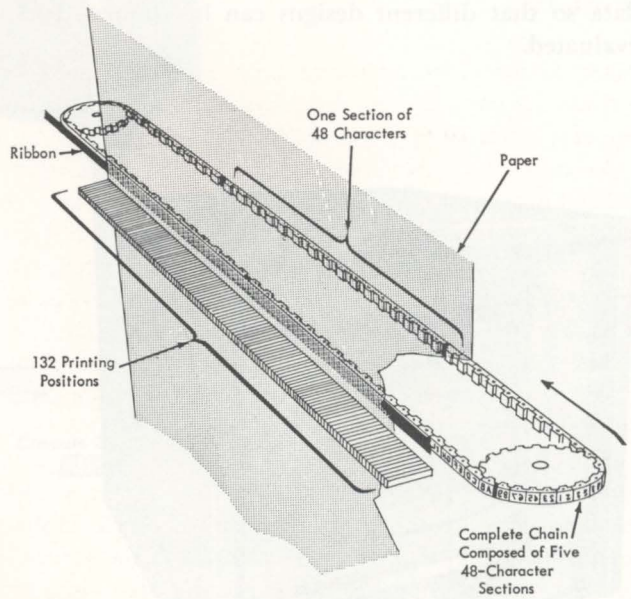


Figure 100. Print Chain

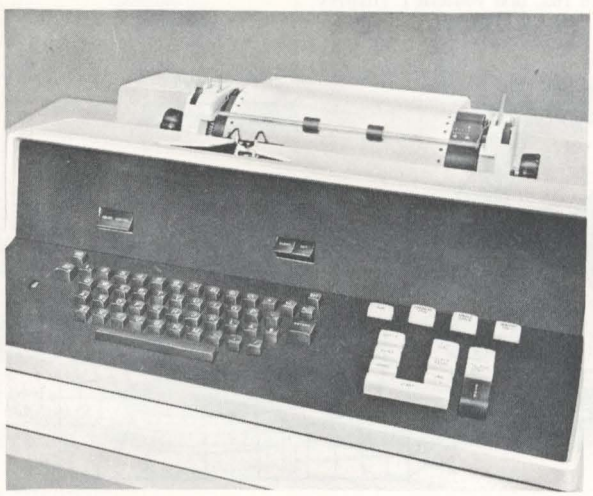


Figure 101. Typewriter on IBM 7150 Console

Cathode Ray Tube

A cathode ray tube display unit provides visual display of processed data on some IBM computer systems. Numbers are converted from digital data to analog data in the form of voltages. These voltages are used to position and control the writing of dots and lines on the face of the cathode ray tube (Figure 102). For example, the actual results obtained by executing a wing stress formula could be used to display the cross-section of a wing design (Figure 103).

A second cathode ray tube with a camera attached is associated with the display unit. This second unit provides a permanent filmed recording of the displayed data so that different designs can be compared and evaluated.

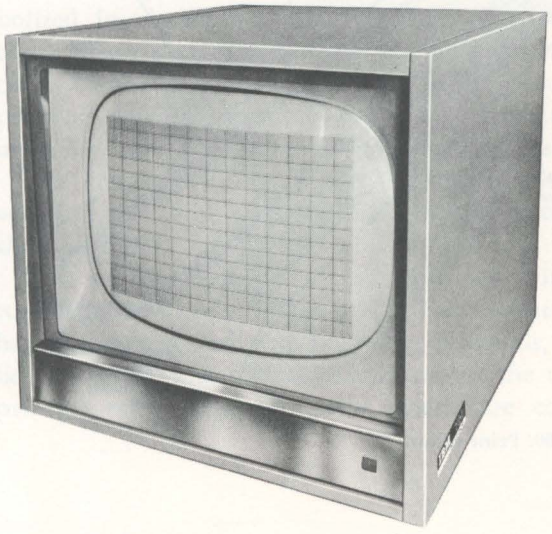


Figure 102. IBM 780 CRT Display

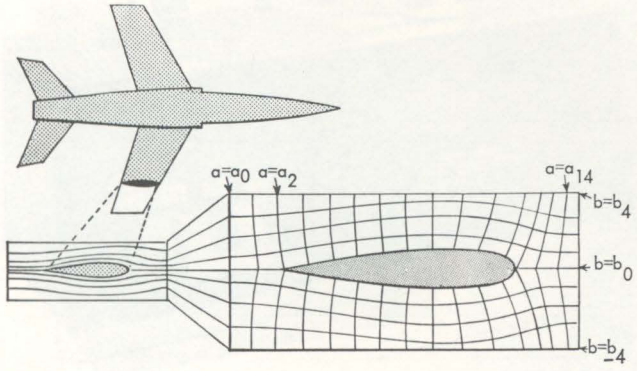


Figure 103. Visual Display of Wing Design

Consoles

The console of a data processing system (Figure 104) is used by the operator to control the system and monitor its operation. Using keys, switches, audible tone signals, and display lights on the console, the operator can:

1. Start and stop the computer.
2. Manually enter and extract (or display) information from internal storage.
3. Determine the status of internal electronic switches.
4. Determine the contents of certain internal registers.
5. Alter the mode of operation so that, when unusual conditions occur, the computer will either stop or indicate the condition and proceed.
6. Change the selection of input-output devices.
7. Reset the computer when error conditions cause it to halt.

In some large data processing systems, the main console is connected only to the central processing unit and may be augmented by separate consoles that are used for engineering functions and for additional input-output control.

Data Buffering

All data processing procedures involve input, processing, and output. Each phase of the procedure takes a specific period of time. The usefulness of a computer is often directly related to the speed at which it can complete a given procedure. Any operation that does not utilize the central processing unit to full capacity prevents the entire system from operating at maximum efficiency. Ideally, the configuration and speed of the various input-output devices should be so arranged that the CPU is always kept busy with useful work.

The efficiency of any system can be increased to the degree in which input, output, and internal data handling operations can be overlapped or allowed to occur simultaneously.

Input is divided into specific units or logical associations of data that enter storage under control of the program. Each unit is normally processed before the next is read in. A number of output results may be developed from a single input, or conversely, several inputs can be combined to form one output result.

Figure 105A shows the basic time relationship between input, processing, and output with no overlap

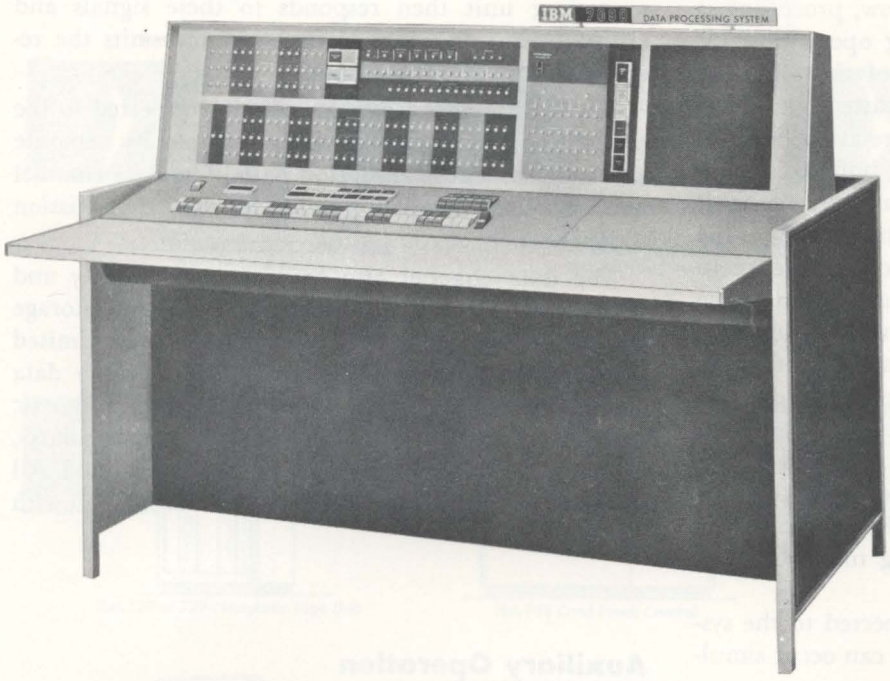


Figure 104. IBM 7151 Console

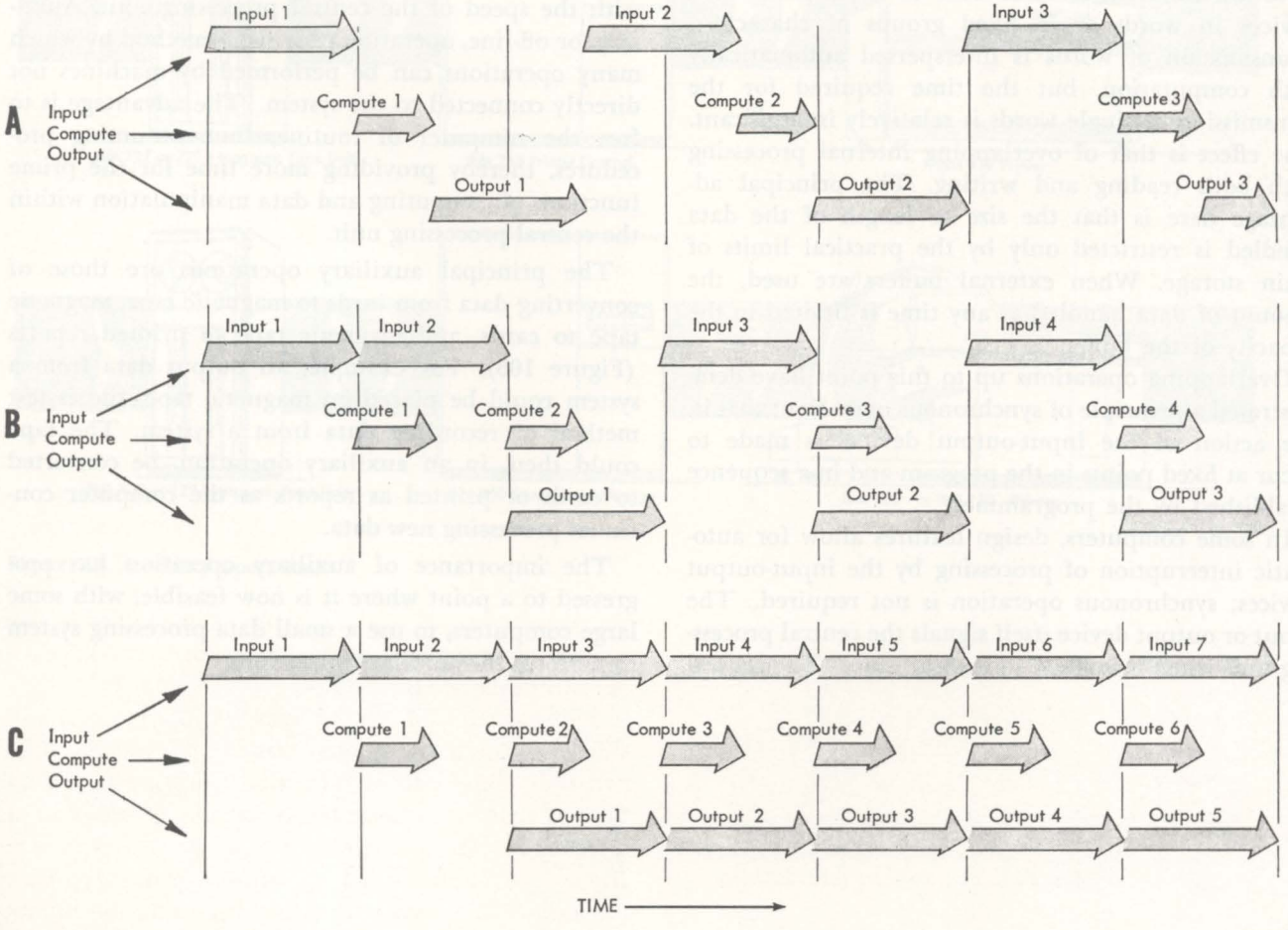


Figure 105. Data Buffering

of operations. In this type of data flow, processing is suspended during reading or writing operations. Inefficiency is obvious, because much of the available time of the central processing unit is wasted.

Figure 105B shows a possible time relation between input-output and computing when a buffered system is used. Data are first collected in an external unit called a buffer. When summoned by the program, the contents of the buffer are transferred to the main storage unit. The transfer takes only a fraction of the time that would be required to read the data directly from an input device. Also, while data are being assembled in the buffer, internal manipulation or computing can occur in the computer. Likewise, completed data from main storage can be placed in the buffer at high speed. The output device is then directed to write out the contents of the buffer. While writing occurs, the central processing unit is free to continue with other work.

If several buffered devices are connected to the system, reading, writing, and computing can occur simultaneously (Figure 105C).

Further development of the buffering concept has led to the use of main storage as the primary buffer. Data are collected from or sent to the input-output devices in words or in fixed groups of characters. Transmission of words is interspersed automatically with computation, but the time required for the transmission of single words is relatively insignificant. The effect is that of overlapping internal processing with both reading and writing. The principal advantage here is that the size or length of the data handled is restricted only by the practical limits of main storage. When external buffers are used, the amount of data handled at any time is limited to the capacity of the buffer.

Overlapping operations up to this point have demonstrated a principle of synchronous operation; that is, the action of the input-output devices is made to occur at fixed points in the program and in a sequence established by the programmer.

In some computers, design features allow for automatic interruption of processing by the input-output devices; synchronous operation is not required. The input or output device itself signals the central processing unit when it is ready to read or write. The central processing unit then responds to these signals and either accepts the data as input or transmits the required information as output.

The input and output devices are connected to the CPU through a data channel, a completely separate and independent information path. The data channel and associated circuitry provide for data transmission independently of computing.

The data channel also controls the quantity and destination of all data transmitted between storage and the input-output devices. It also performs limited counting and testing operations. One or more data channels can be occupied with reading from magnetic tapes while others are writing on other tapes. Similarly, cards may be read or punched and results printed. All of these operations may occur simultaneously with computing.

Auxiliary Operation

Input-output and data conversion operations of the data processing system are relatively slow compared with the speed of the central processing unit. Auxiliary, or off-line, operation provides a method by which many operations can be performed by machines not directly connected to the system. The advantage is to free the computer of routine, time-consuming procedures, thereby providing more time for the prime functions of computing and data manipulation within the central processing unit.

The principal auxiliary operations are those of converting data from cards to magnetic tape, magnetic tape to cards, and magnetic tape to printed reports (Figure 106). For example, all output data from a system could be placed on magnetic tape, the fastest method of recording data from a system. The tape could then, in an auxiliary operation, be converted to cards or printed as reports as the computer continues processing new data.

The importance of auxiliary operation has progressed to a point where it is now feasible, with some large computers, to use a small data processing system to perform the auxiliary operations.

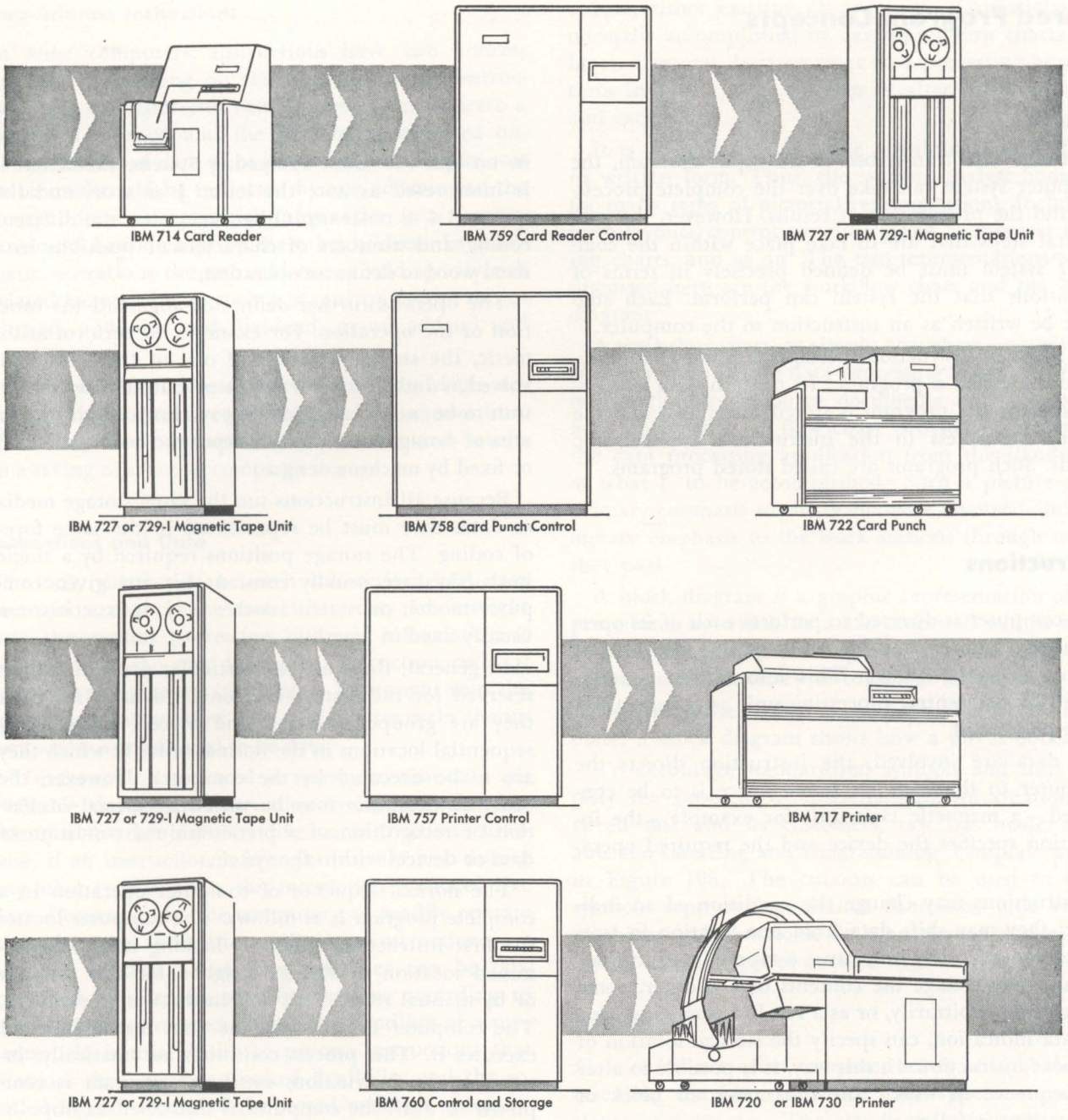


Figure 106. Auxiliary Operations

After data are transcribed to an input medium, the computer system can take over the complete processing and the preparation of results. However, the procedural steps that are to take place within the computer system must be defined precisely in terms of operations that the system can perform. Each step must be written as an instruction to the computer.

A series of instructions pertaining to an entire procedure is called a program. In modern data processing systems the program is stored internally, and the system has access to the instructions at electronic speeds. Such programs are called stored programs.

Instructions

The computer is directed to perform each of its operations by an instruction—a unit of specific information located in main storage. This information is interpreted by the central processing unit as an operation to be performed.

If data are involved, the instruction directs the computer to the data. If some device is to be controlled — a magnetic tape unit for example — the instruction specifies the device and the required operations.

Instructions may change the condition of an indicator; they may shift data from one location in storage to another; they may cause a tape unit to rewind; or they may change the contents of a counter. Some instructions arbitrarily, or as a result of some machine or data indication, can specify the storage location of the next instruction. In this way, it is possible to alter the sequence in which any instruction or block of instructions is followed.

Each instruction (Figure 107) consists of at least two parts:

1. An operation part that designates read, write, add, subtract, compare, move data, and so on.
2. An operand that designates the address of the information or device that is needed for the specified operation.

During an instruction cycle, an instruction is removed from storage and analyzed by the central processing unit. The operation part indicates the operation to be performed. This information is coded to have a special meaning for the computer. For example, in an IBM 705 Data Processing System, the letter G is interpreted as ADD, the letter J as STOP, and the numeral 4 as COMPARE. Other computers use different coding and numbers of characters or positions in a fixed word to define an operation.

The operand further defines or augments the function of the operation. For example, to perform arithmetic, the storage location of one of the factors involved is indicated. For input or output devices, the unit to be used is specified. For reading or writing, the area of storage for input or output records is indicated or fixed by machine design.

Because all instructions use the same storage media as data, they must be represented in the same form of coding. The storage positions required by a single instruction are usually constant for any given computer model; or, stated another way, instructions are usually fixed in length.

In general, there are no particular areas of storage reserved for the instructions only. In most instances they are grouped together and placed in ascending sequential locations in the normal order in which they are to be executed by the computer. However, the order of execution may be varied by special instruction or recognition of a predetermined condition of data or devices within the system.

The normal sequence of computer operation in a complete program is as follows. The computer locates the first instruction either by looking in a predetermined location of storage assigned for this purpose or by manual reset. This first instruction is executed. The computer then locates the next instruction and executes it. This process continues automatically, instruction by instruction, until the program is completed or until the computer is instructed to stop.

Operation	Operand
Select	Tape Unit 200
Read	One Record into Storage Positions 1000-1050
Clear & Add	Quantity in Storage Location 1004 in Accumulator
Subtract	Quantity in Storage Location 1005 from Contents of Accumulator
Store	Result in Storage Location 1051
Branch	To Instruction in Storage Location 5004

Figure 107. Instructions

Two-Address Instructions

In some computers, instructions have two address portions. Depending on the function of the instruction, the two addresses can, for example, indicate a device to be used and the data to be operated on, or two factors of data to be processed. An output unit to be used could be indicated by one address and the storage location from which information is to be written could be indicated by the other address. In arithmetic operations the two addresses could specify two related factors of data such as a multiplier and multiplicand, a divisor and dividend, or an addend and augend.

Fewer double address instructions than single address instructions are required to perform a procedure. This simplifies programming procedures and results in a saving of space in computer storage.

Instructions and Data

The only distinction between instructions and data in main storage is the time when they are brought into the central processing unit. If information is brought into the CPU during an instruction cycle, it is interpreted like an instruction. If brought into the CPU during any another type of computer cycle, the information is considered to be data.

If information is placed in storage so that data are available during an instruction cycle, the computer attempts to treat the data like an instruction. Likewise, if an instruction is brought into the CPU during any other type of cycle, this instruction is treated like data. Consequently, the computer can readily operate upon its own instructions, if those instructions are supplied as data. Also, the computer can be programmed to alter its own instructions according to conditions encountered during the handling of a procedure. It is this ability to process instructions that provides the almost unlimited flexibility and the so-called logical ability of the stored program system.

Developing a Program

To develop a program, the programmer must know: first, the number of different operations available in the system with which he has to work, and their functions; second, and of equal importance, the procedure itself, which must be translated, step by step, into computer instructions; third, the requirements to be met by the result of processing.

The first step in program preparation is a complete analysis of the machine method and the procedure, either existing or proposed. This analysis is normally accomplished by developing flow charts and block diagrams, because most data processing applications involve a large number of alternatives, choices, and exceptions.

It is difficult to state these possibilities verbally or in written form. Thus, the systems analyst finds use for many types of pictorial representations, including form layouts, control panel diagrams, manpower loading charts, and so on. The two representations to be discussed here are the work-flow chart and the block diagram.

A work-flow chart, or simply flow chart, is a graphic representation of the data processing system in which information from source documents is converted to final documents. A flow chart provides a picture of the data processing application from the standpoint of what is to be accomplished. Such a picture gives primary emphasis to the documents involved and secondary emphasis to the work stations through which they pass.

A block diagram is a graphic representation of the procedures by which data are processed within a system. In this picture the emphasis is on the operations and decisions necessary to complete the process.

To summarize: a flow chart shows what job is to be done; a block diagram shows how a job is done.

To encourage standardized symbols and thus simplify the problem of interchanging information between IBM and its customers, IBM has made available the Charting and Diagramming Template shown in Figure 108. The cut-outs can be used to draw symbols representing clerical functions, unit record machines and functions, data processing systems and functions, and types of documents.

Flow Charts

Flow charts are often composed solely of symbols representing the form in which data appear at various stages in a process. The actual operations that must be performed are indicated only briefly or not at all. In other instances, there can be an extension of the data flow chart so that it shows the job steps involved in the development of information as well as the documents themselves.

The symbols in Figure 109 are used in flow charting to illustrate data processing system operations. The rectangle labeled "central processing unit" is used to represent the basic system, including main storage, the arithmetic unit, and the basic controls. To this symbol are connected additional symbols representing other components and the documents that they process. For example, a reel of magnetic tape

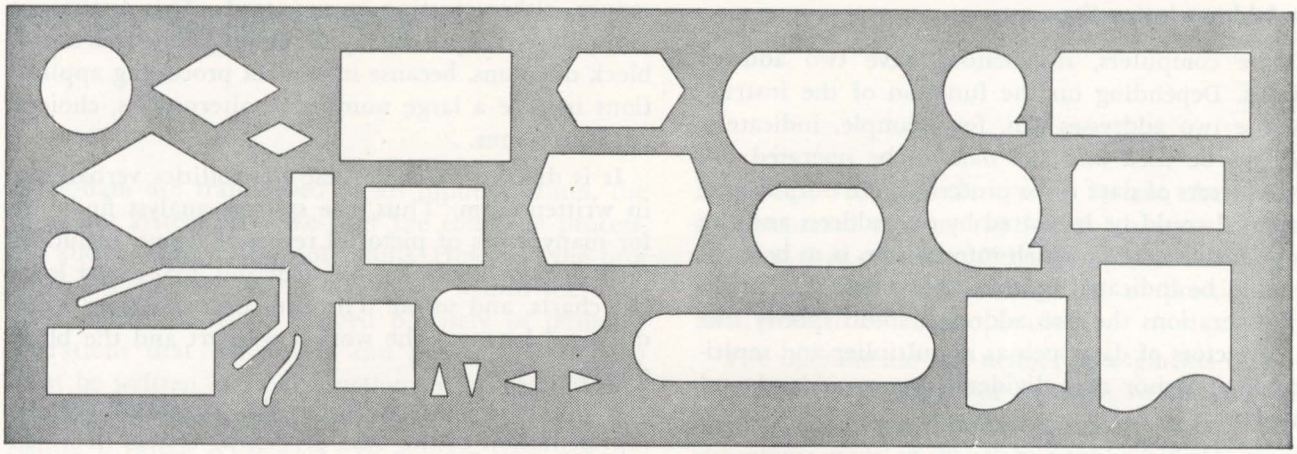


Figure 108. Charting and Diagramming Template

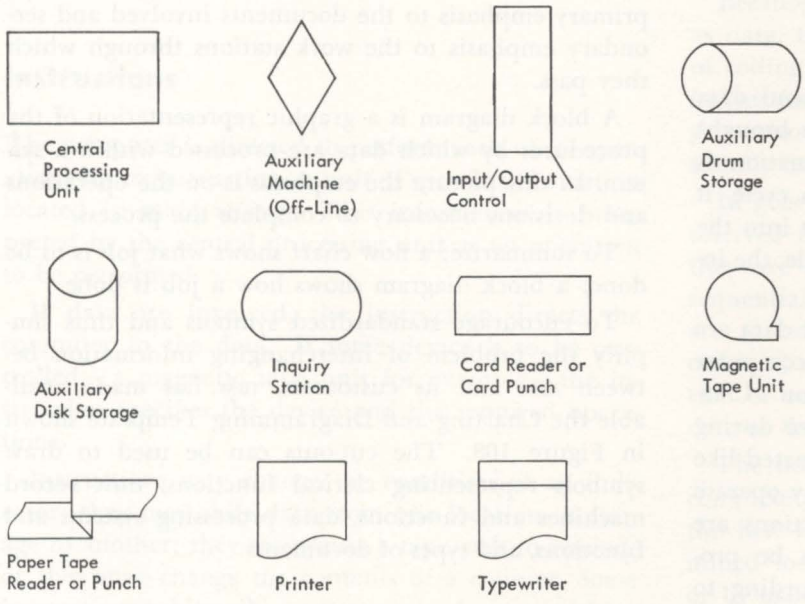


Figure 109. Data Processing System Symbols

indicates that a magnetic tape unit is connected; a punched card with an arrow leading to the CPU indicates a card reader; a punched card with an arrow leading from the CPU, a card punch; and so on.

Figure 110 illustrates typical ways in which data processing system operations are flow-charted. The central processing unit symbol is sometimes labeled with the type number to identify the system. When no confusion about the system type can exist, the space can be used to identify the processing step.

Figure 111 is a typical operational flow chart for an application of a data processing system—in this case the IBM 705. Note that the primary direction of flow of the application is in a vertical line from the top of the page to the bottom. Secondary functions are shown to one side.

The labeling of an operational flow chart for a data processing system application is important when it is to serve as a guide to programming and operating.

Block Diagrams

The symbols in Figure 112 are used in block diagramming to illustrate the data processing procedures.

One of the most important uses of the block diagram is to provide the programmer with a means of visualizing, during the developmental stages of programming, the sequence in which logical and arith-

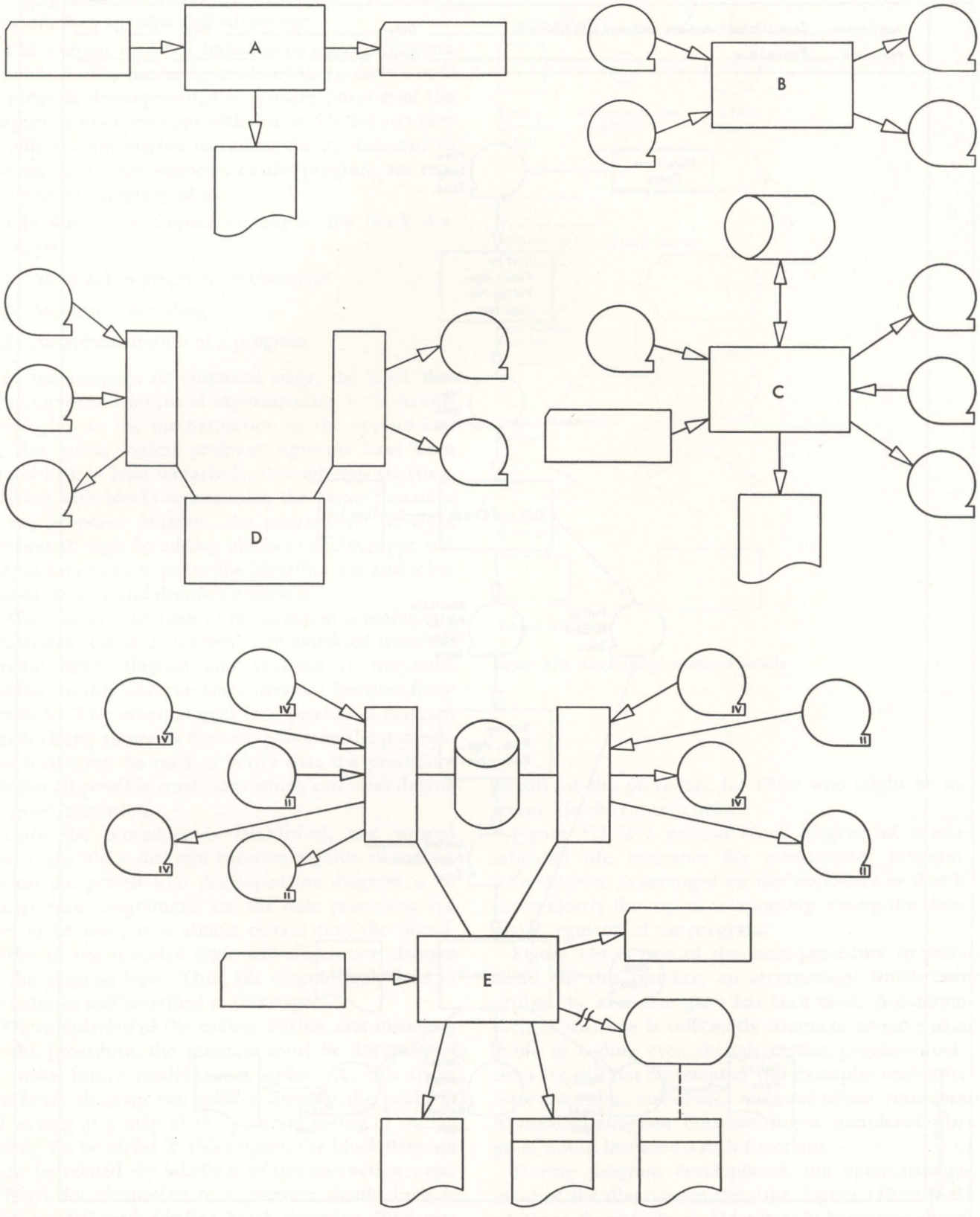


Figure 110. Typical Data Processing System Configurations

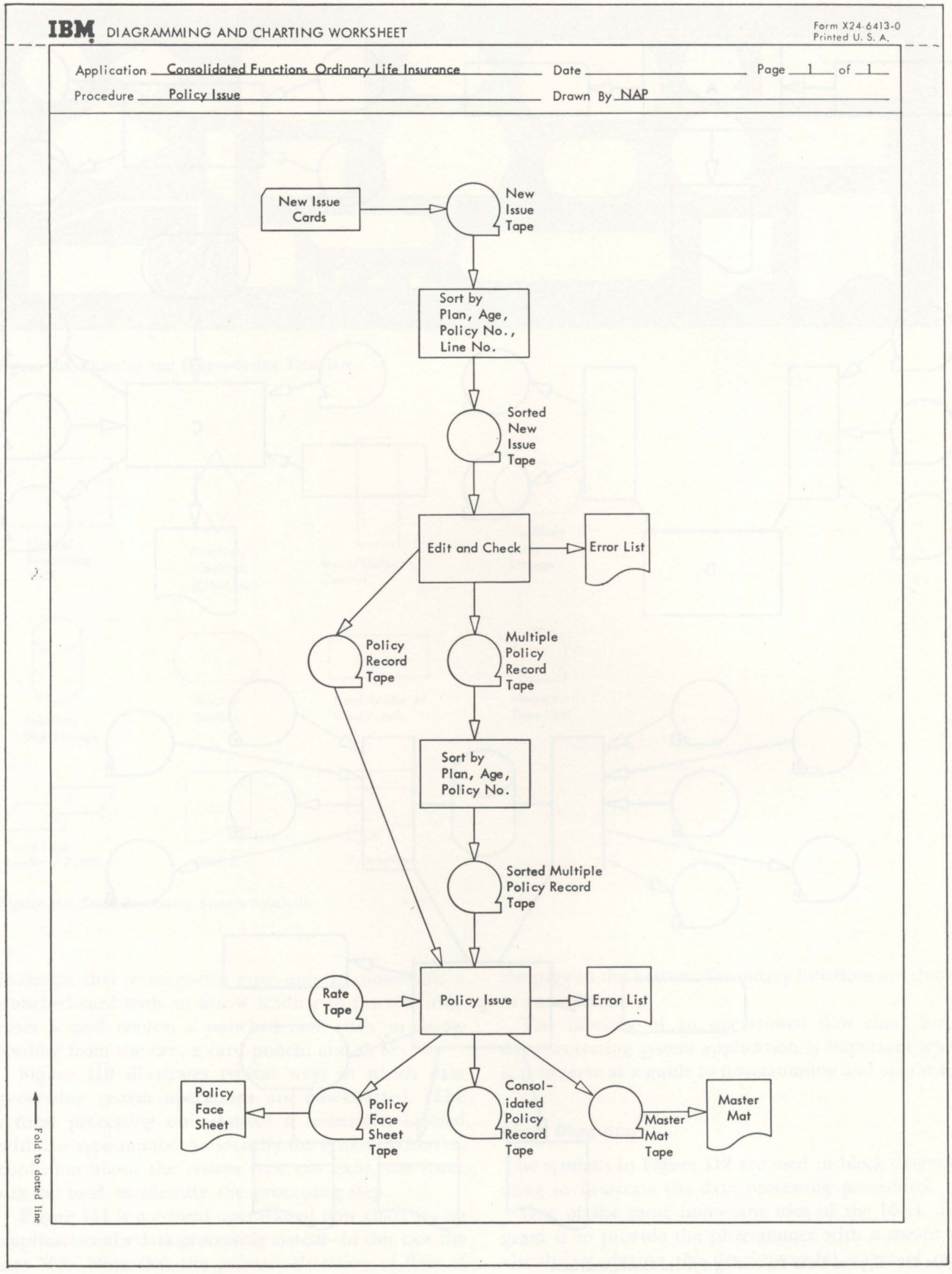


Figure 111. Operational Flow Chart

metic operations should occur, and the relationship of one portion of a program to another. A written program does not offer this advantage.

The amount of detail included in a block diagram depends on the use to be made of it. In early stages of program development, the primary purpose of the diagram is to experiment with and verify the accuracy of different approaches to coding the application; in this instance large segments of the program are represented by a single symbol.

The three most important uses of the block diagram are:

1. As an aid to program development
2. As a guide to coding
3. As documentation of a program

In the program development stage, the block diagram serves as a means of experimenting with various approaches to the mechanization of the application. At this point, logical program segments have been established, at least tentatively, through flow charting. Starting with blocks representing the major functions of the proposed program, the programmer develops the over-all logic by adding blocks to depict input and output functions, steps for the identification and selection of records, and decision functions.

After the over-all logic of the program is tentatively established, the large segments are extracted from the general block diagram and analyzed in the same fashion. In this way, the block diagram becomes more detailed. The eventual goal is to produce a diagram which clearly shows all decision points in the program and which can be used to verify that the procedure satisfies all possible conditions which can arise during program execution.

Once the procedure is established, and proved sound, the block diagram becomes a guide to coding. Unless the person who developed the diagram is an experienced programmer for the data processing system to be used, it is almost certain that the peculiarities of the machine logic will necessitate changes in the program logic. Thus, the diagram may need to be redrawn and reverified at this stage.

On completion of the coding, testing, and installing of the procedure, the program must be documented to make future modifications easier. At this stage, the block diagram can greatly simplify the problem by serving as a map of the program listing of coding sheets. To be useful in this respect, the block diagram must be related, by labeling, to the instruction steps.

Final documentation of a program should include both general and detailed block diagrams. The general block diagram helps in understanding the more detailed ones and also provides an easily understood

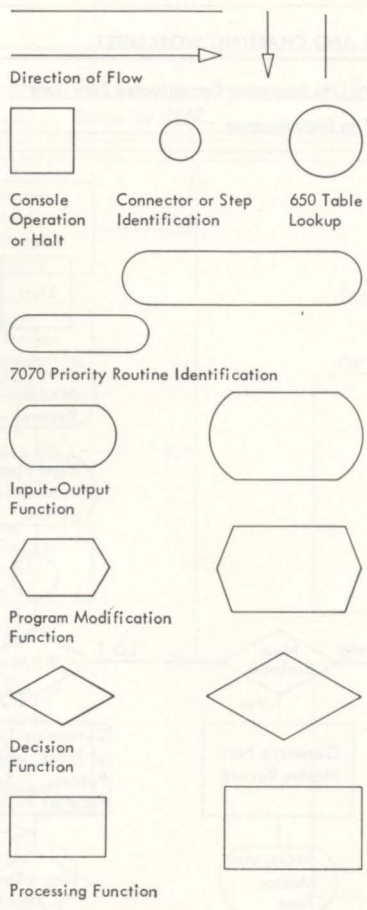


Figure 112. Block Diagramming Symbols

picture of the procedure for those who might be interested in the concept only.

Figure 113 is a general block diagram of a consolidated life insurance file maintenance program. This diagram is arranged on the worksheet so that it shows clearly the logical relationship among the three major segments of the program.

Figure 114 is part of the same procedure in more detail. In this instance, an arrangement which best utilizes the available space has been used. A diagram such as this one is sufficiently complete to serve as a guide in coding even though certain program functions are still not represented (for example: read-write error routines, end-of-file, and end-of-job routines). A detailed diagram documenting a completed program would include all such functions.

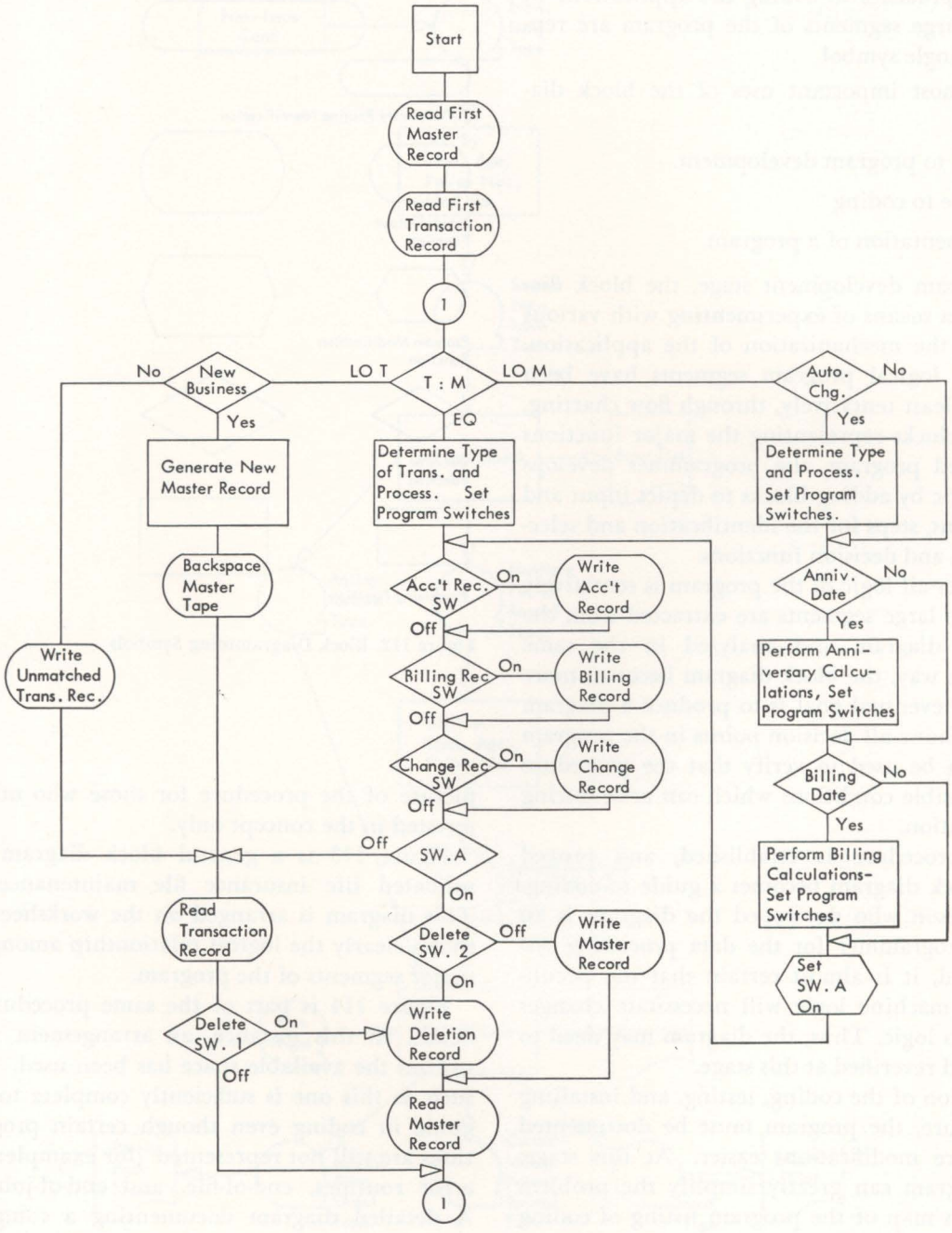
During program development, the open arrangement of the diagram on the page, Figure 113, is preferable to that of Figure 114 not only because it shows more clearly the alternate program paths, but also because it leaves more room for inserting additional

IBM DIAGRAMMING AND CHARTING WORKSHEET

Form X24-6413-0
Printed U. S. A.

Application Ordinary Life Insurance Consolidated Functions Date _____ Page 1 of 1

Procedure Daily File Maintenance Drawn By NAP



Fold to dotted line

Figure 113. General Block Diagram

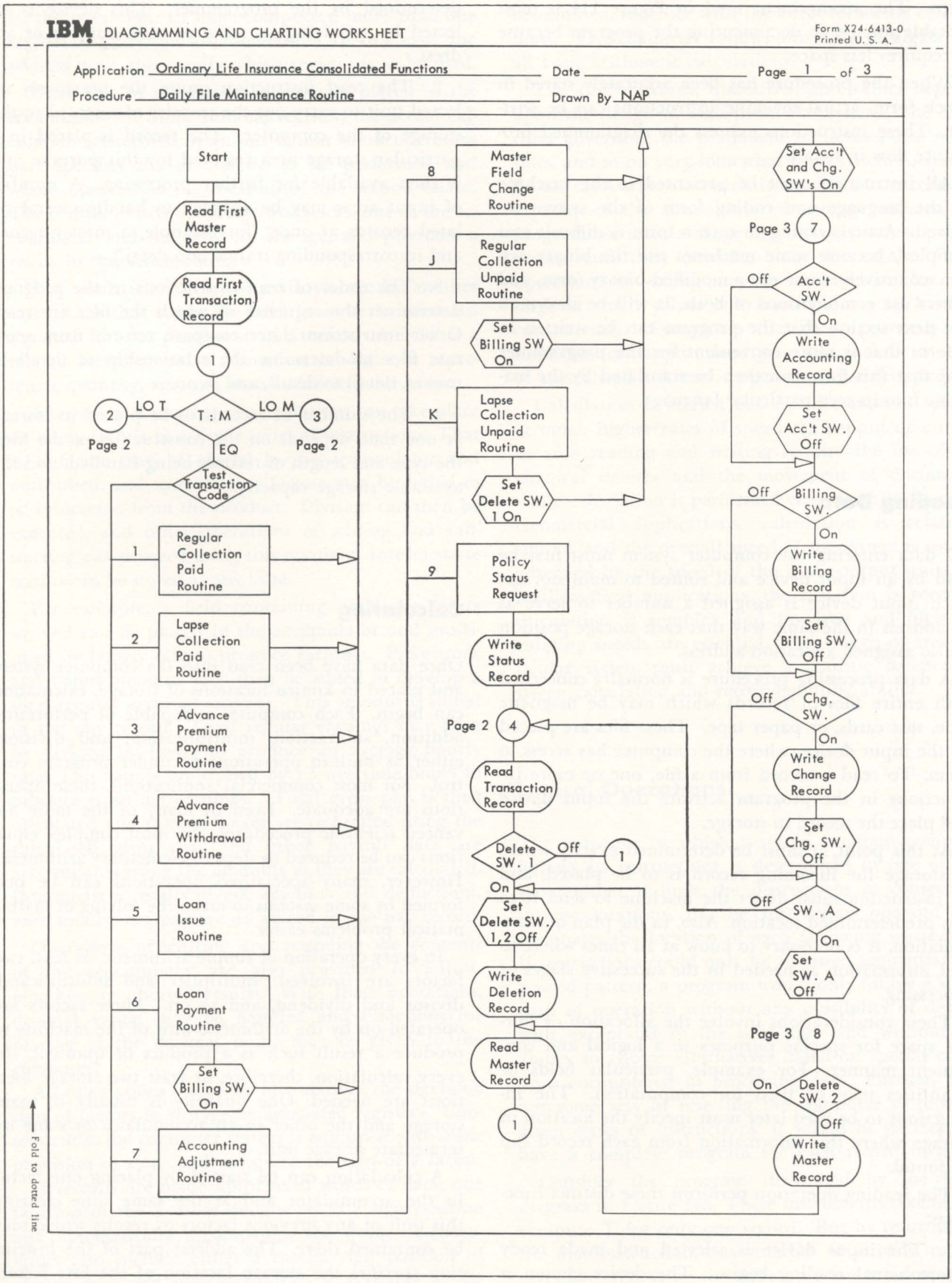


Figure 114. Detailed Block Diagram

steps. The arrangement used in Figure 114 is most suitable for use in documenting the program because it requires less space.

When the procedure has been accurately stated in block form, actual machine instructions can be written. These instructions express the diagrammed procedure flow in detail.

All instruction must be presented to the machine in the language and coding form of the system involved. Actual writing in such a form is difficult and complex, because some machines use the binary system exclusively, some use a modified binary form, and others use combinations of both. It will be shown in the next section that the program can be written in a form that is more convenient for the programmer and that this form can then be translated by the machine into its own particular language.

Reading Data

All data entering the computer system must first be read by an input device and routed to main storage. Each input device is assigned a number to serve as its address in the same way that each storage position is also assigned a location address.

A data processing procedure is normally concerned with entire files of records which may be magnetic tape, IBM cards, or paper tape. These files are placed on the input device, where the computer has access to them. To read a record from a file, one or more instructions in the program activate the input device and place the record in storage.

At this point, it must be determined exactly where in storage the incoming record is to be placed, and an instruction must direct the machine to send it to this predetermined location. Also, in the plan of manipulation, it is necessary to know at all times where to find information as needed in the successive stages of processing.

These considerations involve the allocation of storage space for specific purposes in a logical and convenient manner. For example, particular fields or quantities may be used for computation. The instructions to be used later must specify the location in storage where this information from each record can be found.

The reading operation performs these distinct functions:

1. The input device is selected and made ready before actual reading begins. The device chosen is the one that has access to the proper file of records as determined by the programmer. This device is selected by specifying its assigned code number or address.

2. The read instruction causes the previously selected unit to carry out the transfer of a record to the storage of the computer. The record is placed in a particular storage area reserved for this purpose and is then available for further processing. A number of input areas may be assigned to handle several related records at once (for example, a master record and its corresponding transaction detail).

3. The order of read instructions in the program determines the sequence in which the files are read. Other instructions later compare records from separate files to determine the relationship of detail to master, detail to detail, and so on.

4. The number of records to be placed in storage at one time depends on the construction of the files, the type and length of records being handled, and the available storage capacity.

Calculating

Once data have been read into the computer system and placed in known locations of storage, calculation can begin. Each computer is capable of performing addition, subtraction, multiplication, and division, either as built-in operations or under program control. For most commercial applications, these operations are adequate. Even in many of the more advanced scientific procedures, the most complex equations can be reduced to steps of elementary arithmetic. However, many specialized operations can be performed by some systems to make the solving of mathematical problems easier.

In every operation of simple arithmetic, at least two factors are involved: multiplier and multiplicand, divisor and dividend, and so on. These factors are operated on by the arithmetic unit of the machine to produce a result such as a product or quotient. In every calculation, therefore, at least two storage locations are needed. One quantity is usually in main storage and the other in an accumulator or some intermediate storage unit.

A calculation can be started by placing one factor in the accumulator and at the same time clearing this unit of any previous factors or results which may be contained there. The address part of the instruction specifies the storage location of the first factor; the accumulator or other storage register is implied

by the operation. In some computers, more than one register is available for calculation. In this case, the address must also specify which register is to be used.

When one of the factors is properly placed in the accumulator or other suitable register, the actual calculation is executed by an instruction whose operation part specifies the arithmetic to be performed and whose operand is the location of the second factor. The computer acts upon the two factors, one in the accumulator and the other in storage, and produces a result in the accumulator.

The result is returned to main storage by another instruction that stores the field in some location designated for the placement of a result. A field is a related arrangement of characters or digits to represent a quantity, amount, name, identity, and so on.

Any practical number of calculations can take place on many factors in a single series of instructions. That is, a factor may be placed in the accumulator and multiplied, and several other factors may be added to or subtracted from the product. Division can then be executed, and other operations of adding and subtracting can proceed using this quotient. Intermediate results can be stored at any time.

For example, a field containing employee hours worked can be placed in the accumulator and multiplied by hourly rate to produce earnings. Piece work and bonus amounts may then be added to develop a total regular earnings amount. This amount is stored in the pay record. Total regular earnings may then be divided by hours to produce an average hourly rate. This rate is multiplied by 1.5 overtime hours to produce overtime earnings. Total gross pay is then calculated and stored. Taxes are computed using the calculated gross pay, and other payroll data are accumulated using the amounts as they are calculated. Intermediate results of tax amounts and deductions, and, finally, net pay are all stored in the pay record.

Operations of shifting and rounding the contents of the accumulator are also provided to adjust, lengthen, or shorten results. With these operations, decimal values may be handled and directions for placing of the decimal point may be given to the computer.

All calculations must take into account the algebraic sign of factors in storage or associated registers. Consequently, the computer system is equipped with some provision to store and recognize the sign of a factor.

If records are made up of fixed words of data, one position of the word is designated as a sign position and automatically accompanies the word. Accumulators also include either a special sign position of storage or a sign indicator which is available to the programmer. In this way, the sign of results can be determined, together with the effect following calculations. The computer follows the rules of algebra in all basic arithmetic calculations.

The size of words, quantities, and values depends upon the design of each particular system. The exact rules governing the placement of factors, size of results, and so on vary somewhat from system to system. In all cases where a result is expected to exceed the capacity of the accumulator or storage register, the programmer must arrange his data to produce partial results and then combine these for totals. Other operations of scaling may be executed so that very large or small values and fractions may be handled conveniently. Computers designed primarily for mathematical applications generally include a series of specialized arithmetic operations for this purpose.

Calculation is carried out in all computer systems at much higher rates of speed than input or output, because reading and writing require the use of mechanical devices and the movement of documents, while calculation is performed electronically. In many commercial applications, calculation is relatively simple, and the over-all speed of the system is usually governed by the speed of the input-output units. In mathematical applications, the situation is reversed; calculation is complex and involved and high calculating speeds are essential. The design of any particular system must achieve a realistic balance between calculating and record-handling ability.

Logical Operations

The sequence in which a stored program computer follows its instructions is determined in one of two ways: either it finds the instructions in consecutive storage locations, or the instruction operand also designates the location of each following instruction. If instructions could only be followed sequentially in a fixed pattern, a program would only follow a single path of operation without any possibility of dealing with exceptions to the procedure and without any ability to choose alternatives based on special conditions encountered in processing data. Further, without some way of resetting the computer to repeat a given series of instructions, it would be necessary to have a complete program for each record in a file.

Consider the program illustrated by the block diagram in Figure 115. These instructions taken alone compute T for only one record. But by returning to the first instruction, any number of records may be processed, repeating the same program as a loop. For this purpose, another instruction is given to return to

the first instruction (Figure 116). Once this program is initiated, it will continue to run until there are no more records to process. Program loops are common and they can be terminated in many ways.

For example, the computer may be instructed to examine T each time it is computed and to stop when the value of T becomes negative (Figure 117). In this case, the instruction becomes a conditional transfer. The program is repeated only if some predetermined condition has been satisfied. The computer can also be instructed to execute the program for ten records and then stop (Figure 118). It is assumed that the constants 10 and 1 are in the computer and that 1 is subtracted from 10 each time the loop is completed. After ten times around, a zero will be in the location that contained 10. A transfer or branch then terminates the loop.

The conditional transfer or branch operation may be used to cause a special-purpose subroutine to be executed outside the normal or straight-line path of the program. This subroutine is executed only when a predetermined exception or condition is noted by the machine.

One common example of the subroutine is checking the accuracy of records as they are read from or written on magnetic tape. As each record enters or leaves the central processing unit, a read-write error indicator is examined. If the indicator has been turned on, the computer is instructed to enter a subroutine of instructions that attempt to correct the error. The program logic for such an operation—the reading only

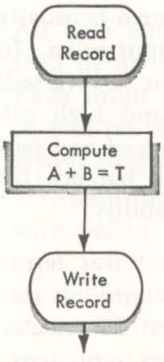


Figure 115. Block Diagram, A + B = T

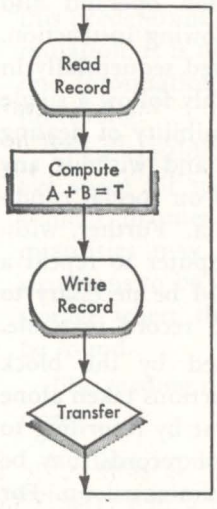


Figure 116. Program Loop

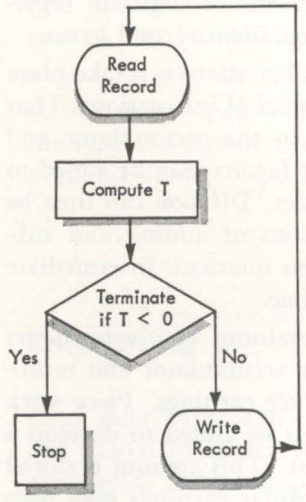


Figure 117. Conditional Transfer

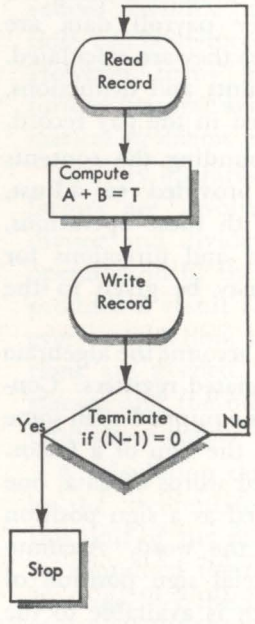


Figure 118. Record Count Conditional Transfer

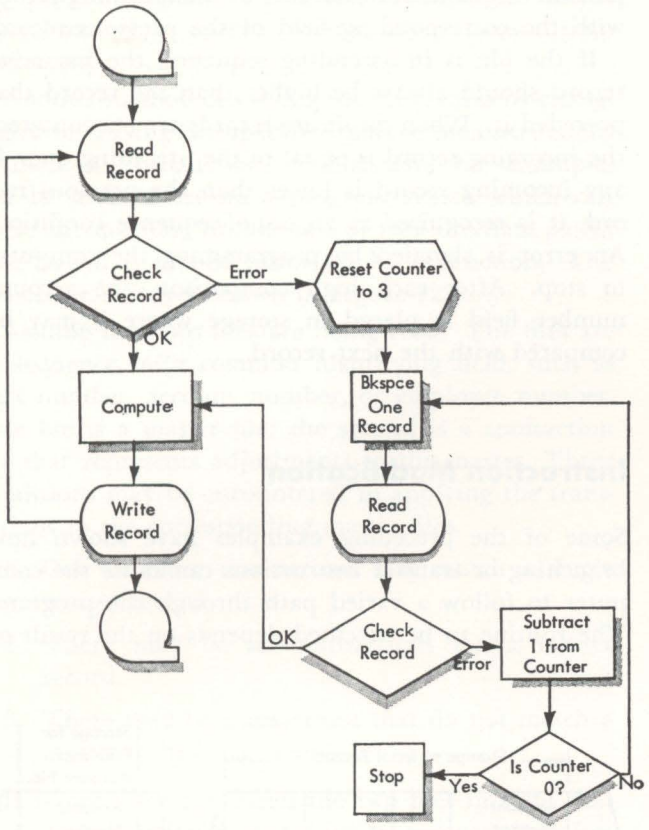


Figure 119. Read Error Loop

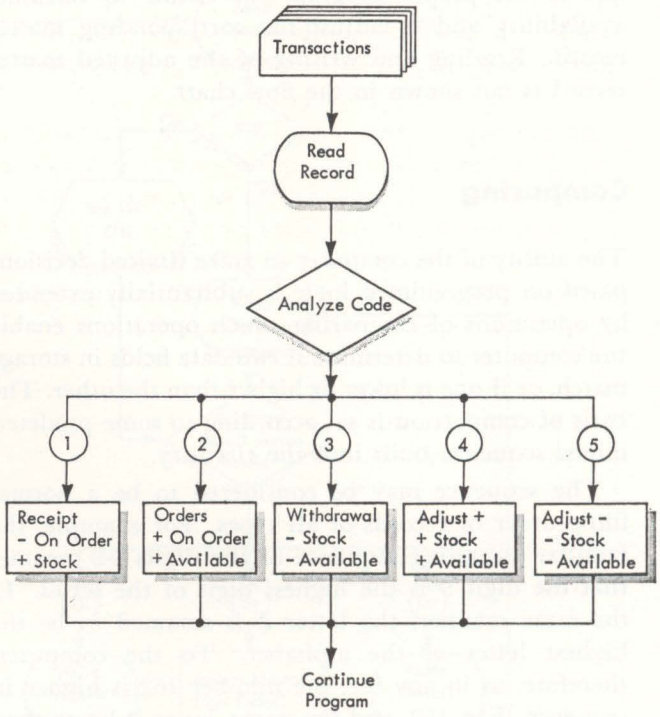


Figure 120. Branching By Code

—is shown in Figure 119. A similar loop might also be included for writing.

When a reading error is detected, a transfer is effected to the error subroutine. A counter is reset to the quantity 3 to count the number of times a re-read will be attempted. The tape is backspaced over the error, and a second read instruction is given. Another check is made to determine if this operation is correct. If it is, a transfer returns to the main program, where computing continues.

If the error persists, 1 is subtracted from the counter and the counter is tested for 0. The error loop is again entered and a second re-read and check are executed. The machine can re-read three times. If the error is not corrected, operation is halted. Further instructions can also be programmed to indicate to the operator the cause of the stop.

A program can also be arranged so that the machine can recognize one or more types of records as they are processed from a single file. The method of computation can be varied according to the type of record in storage. This procedure is common when a number of types or classes of transactions are processed against a single master file (for example, in an application of file maintenance).

Assume that a file of master stock status records contains quantities that reflect the number of parts available for manufacturing planning. The records also have considerable other information pertaining to the status of inventory, but for purposes of illustration, this example is concerned only with those fields used to show availability. These fields are:

Quantity in stock
Quantity on order
Quantity available

Transactions that affect the status of the parts availability originate daily. These transactions are punched in cards with an identifying digit code for each type of activity.

Codes are as follows:

Code 1	Receipts
Code 2	Orders
Code 3	Withdrawals
Code 4	Adjustments plus
Code 5	Adjustments minus

As each transaction is placed in storage, it is analyzed by code to determine in which class it belongs (Figure 120). A branch instruction then trans-

fers to the proper program subroutine to calculate availability and to adjust the corresponding master record. Reading and writing of the adjusted master record is not shown in the flow chart.

Comparing

The ability of the computer to make limited decisions based on programmed logic is substantially extended by operations of comparing. Such operations enable the computer to determine if two data fields in storage match, or if one is lower or higher than the other. The basis of comparison is set according to some predetermined sequence built into the circuitry.

The sequence may be considered to be a normal filing order of records of all types. For example, the familiar ascending sequence of the digits 0-9 assumes that the digit 9 is the highest digit of the series. In the same manner, the letter Z is assumed to be the highest letter of the alphabet. To the computer, therefore, as in any file, the number 162 is higher in sequence than 159, and the name Jones is lower than the name Smith. Special characters, such as / @ * , or − , may also be included because these characters must be manipulated as data for report printing and other special purposes.

Comparing operations are used to program the sequence checking of files, sorting procedures, or the rearrangement of records in some desired order. The comparison of an identifying field in one record with that of another enables the computer to handle a number of associated files in one processing procedure provided that all files are in sequence by this common field.

The two fields to be compared are always in main storage. One field is then placed in an accumulator or storage register, and a compare instruction is given to compare this field against the second specified field which remains in main storage. The results of comparison are registered as high, low, or equal, by indicators or triggers which may then be interrogated to determine their condition. If the indicator is on, a branch instruction transfers the machine to a subroutine which will continue processing according to the result of the comparison.

Figure 121 shows a typical program arrangement for sequence checking a single file of records. All records in the file are assumed to be in ascending sequence by account number. An input area is set aside in storage where records are received, one at a time, from an input unit. A second area is also reserved in storage to store the account number from the preceding record. The purpose of this area is to allow comparison of the account number of the incoming record with the corresponding field of the previous record.

If the file is in ascending sequence, the incoming record should always be higher than the record that preceded it. When duplicate records are encountered, the incoming record is equal to the preceding one. If any incoming record is lower than the previous record, it is recognized as an out-of-sequence condition. An error is signaled by programming the computer to stop. After each high comparison, the account number field is placed in storage where it may be compared with the next record.

Instruction Modification

Some of the preceding examples have shown how branching or transfer instructions can cause the computer to follow a varied path through the program. The routine to be executed depends on the result of

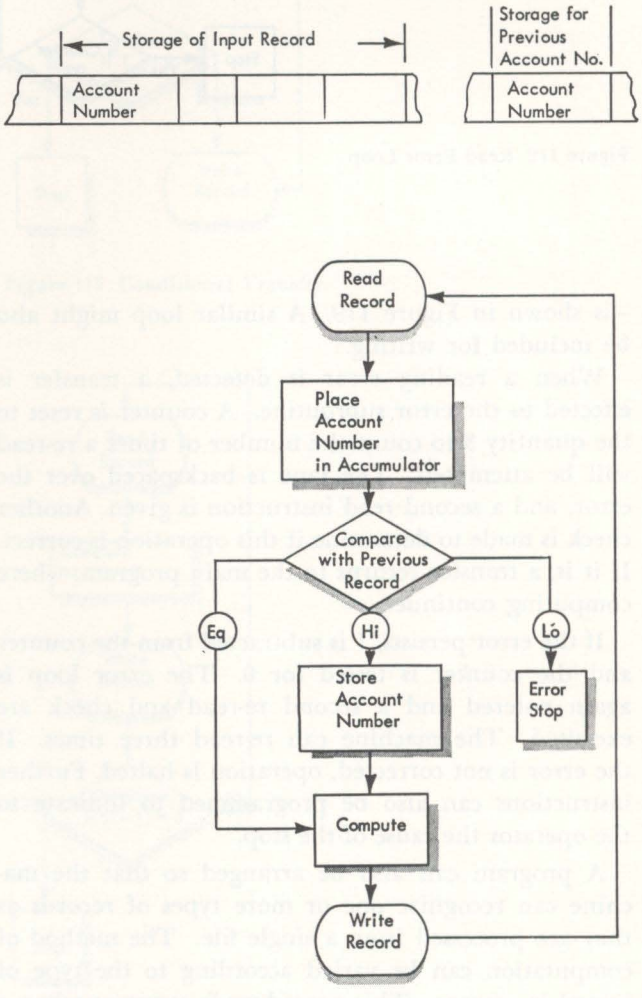


Figure 121. Sequence Checking

a previous comparison or a test of indicators which have been set by a zero balance, an error condition, and so on.

Another method of varying the program is by changing or modifying the operation part of the instructions themselves. Instruction modification, for example, can be used to set up a *program switch* which can cause the machine to take one of two alternate paths. The switch is turned on or off by instruction. The use of the switch is shown in Figure 122.

Assume that two files are being read. The files are in sequence by a common identifying field, such as part number, account number, or employee number. One file is a master file; the second is a transaction file that represents adjustments to the master. Three conditions may be encountered in applying the transactions to the corresponding master files.

1. One or more transactions may match a single master record.
2. There may be no transactions for a master record.
3. There may be transactions that do not match a master — these are errors.

It is necessary to process the two files in step; that is, each transaction record must be compared against a corresponding master record, if there is one. If several transactions apply to the same master record, the transaction file must continue reading without reading a new master record. Conversely, if a master record is read in without a corresponding transaction, this record is written out unchanged and the following master is read in. The reading and writing of master records continue until a matching transaction is found.

The flow chart shows that one master record is read in first. A switch instruction is inserted between the reading of the master and the transaction. As operations begin, this switch is turned off, allowing one transaction to be read in. The identifying field of the transaction is compared against the master. If it is equal, the master is adjusted and a second transaction is read in. If this transaction is not to be applied against the master (which is still in storage), it should be high when compared. The previously adjusted master is then written out and the switch is turned on. A new master is then placed in storage, but because the switch is on, a transaction is not read; instead the machine transfers directly to the instruction to compare. The switch is turned off each time this happens. Operation continues with comparison for each new record placed in storage. If a transaction is low, it is written out on a separate output unit, and a new transaction is then read in.

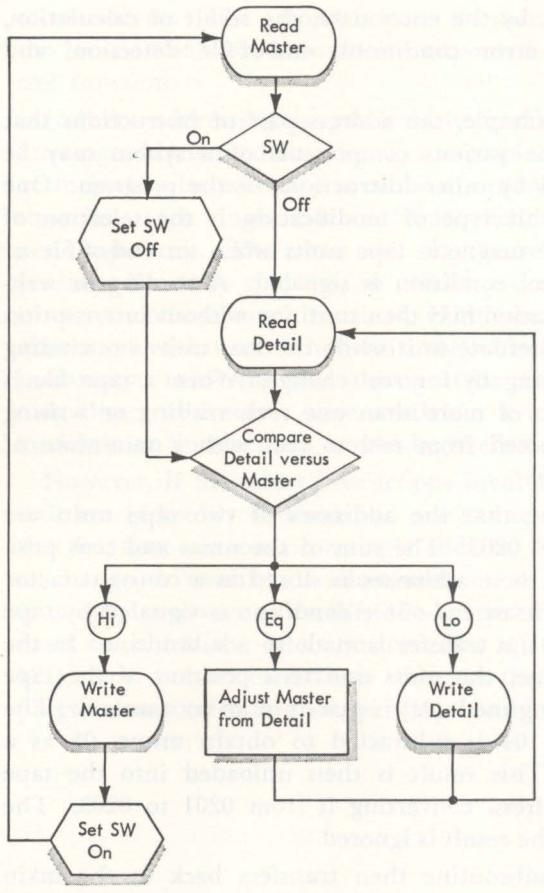


Figure 122. Program Switch

The switch, when on, has an operation part specifying an unconditional transfer. The address part is the location of the compare instruction. To turn the switch off, the operation part is changed to no operation. In this case, the machine ignores the instruction and proceeds to the following instruction: read a transaction.

Address Modification

The address portion of instructions may also be treated as data. An instruction address can be modified by arithmetic; it may be compared against other addresses or factors or relocated in storage at will. Address modification serves two purposes:

1. The total number of instructions in a program may be reduced, conserving storage capacity for data or other factors. One instruction, or a single series of instructions, can serve to address variable locations in storage.

2. A basic flow of work controlled by the program can serve as a pattern of procedure that can change as

required by the entry data, the result of calculation, various error conditions, end-of-file detection, and so on.

For example, the address part of instructions that select the various components of a system may be modified by other instructions in the program. One use of this type of modification is the selection of alternate magnetic tape units when an end-of-file or end-of-reel condition is signaled. A reading or writing operation may then continue without interruption on an alternate unit while the first unit is rewinding or standing by for reel change. When a tape file is made up of more than one reel, reading or writing may proceed from reel to reel with a minimum of lost time.

Assume that the addresses of two tape units are 0201 and 0203. The sum of the units and tens positions of these addresses is stored as a constant factor 04. When an end-of-file condition is signaled by tape unit 0201, a transfer is made to a subroutine. In the subroutine, the units and tens position of the tape unit being used (01) is placed in an accumulator. The constant 04 is subtracted to obtain minus 03 as a result. This result is then unloaded into the tape unit address, converting it from 0201 to 0203. The sign of the result is ignored.

The subroutine then transfers back to the main routine and continues to use tape unit 0203. When end-of-file is signaled from this unit, the constant 04 is subtracted from 03 to obtain the result minus 01. Unloading this result changes the tape address from 0203 to 0201. The select address alternates between 0201 and 0203 each time an end-of-file is signaled. Similar factors may be used to form other types of program alternators.

Indexing

In many computers, the address portion of an instruction can be modified by adding or subtracting variable quantities contained in one or more special-purpose counters. The counter may be called an index register when it is set aside specifically for this purpose or it may be a predetermined location in core storage called an index word. A computer may have several index registers or a number of storage locations for index words. Both the index register and the index word perform identical functions; however, the word is usually more accessible to the program and, consequently, offers more flexibility in its use.

Computers with an indexing feature use an expanded instruction format that allows a particular register or word to be specified as a part of the instruction operand.

Assume that fifty quantities are placed in ascending word positions of storage from locations 1001 to 1050 inclusive and that these quantities are to be added to the contents of an accumulator. Without indexing or address modification, it is necessary to repeat an add instruction fifty times with the address of each instruction incremented by 1, as ADD 1001, ADD 1002, ADD 1003, and so on.

With indexing, the add instruction can be written as ADD 1051 with the address decremented by an index register containing the quantity 50. The address remains 1051, but the computer calculates an effective address of 1051 minus 50, or 1001. When the add instruction is executed, the contents of the index register are also decremented by 1, leaving a remainder of 49. When the same add instruction is re-executed and is again decremented by the contents of the same index register, the effective address is 1051 minus 49, or 1002. If a program loop is formed to repeat this process, the effective address of the add instruction is stepped up 1 each time it is executed (as the index register is stepped down). When the index register equals 0, all 50 quantities will have been added and the loop is terminated. The computer has consequently performed fifty operations using the same instructions.

Figure 123 is a flow diagram of the index loop. The first instruction places the quantity 50 in index register 3. An add instruction, with an address 1051, also specifies as part of its operand a designation that

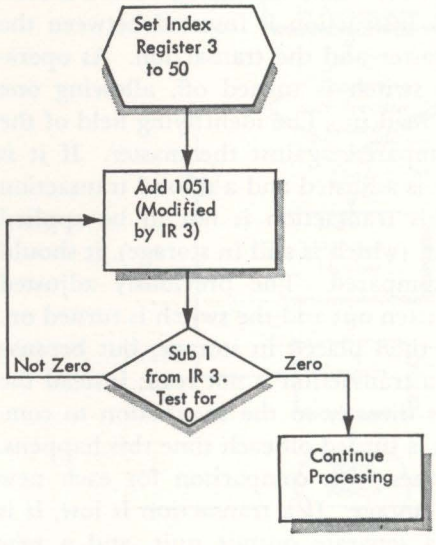


Figure 123. Index Loop

the given address is to be modified by the quantity contained in index register 3. The next instruction is branch on index, which means: reduce the contents of index register by 1; if the contents of the register are greater than zero, branch to repeat the add instruction; if the contents of the index register equal zero, continue to the next instruction in the program.

The indexing feature greatly simplifies programming of repetitious calculations or other operations and reduces the required number of instructions.

Indirect Addresses

All instruction addresses discussed in preceding illustrations are classified as direct, that is, they refer directly to the location of data or other instructions in storage, they select a machine component, or they specify the type of control to be exercised.

Addresses may also be indirect. Such an address can refer only to a storage location that contains another address. The second address in turn refers to the location of data, a machine component, or a control function.

Indirect addressing is particularly useful in performing address modification. For example, in a program it may be necessary to address a number of instructions to a single machine unit, such as a magnetic tape unit. The same unit may be selected for reading, for error routines, in restart procedures, or in various other branch routines. If this unit is to be alternated with some other unit, the addresses of all instructions involving that unit must be modified. If only direct addressing were available, a number of modification instructions would be needed.

However, if the select instructions involving a tape unit are indirectly addressed to one storage location, that location can contain a single address. Therefore, to change or modify all select instruction addresses, it is only necessary to modify the single effective address to which the select instructions refer (Figure 124). Any number of indirect addresses throughout a program may refer to a single effective address.

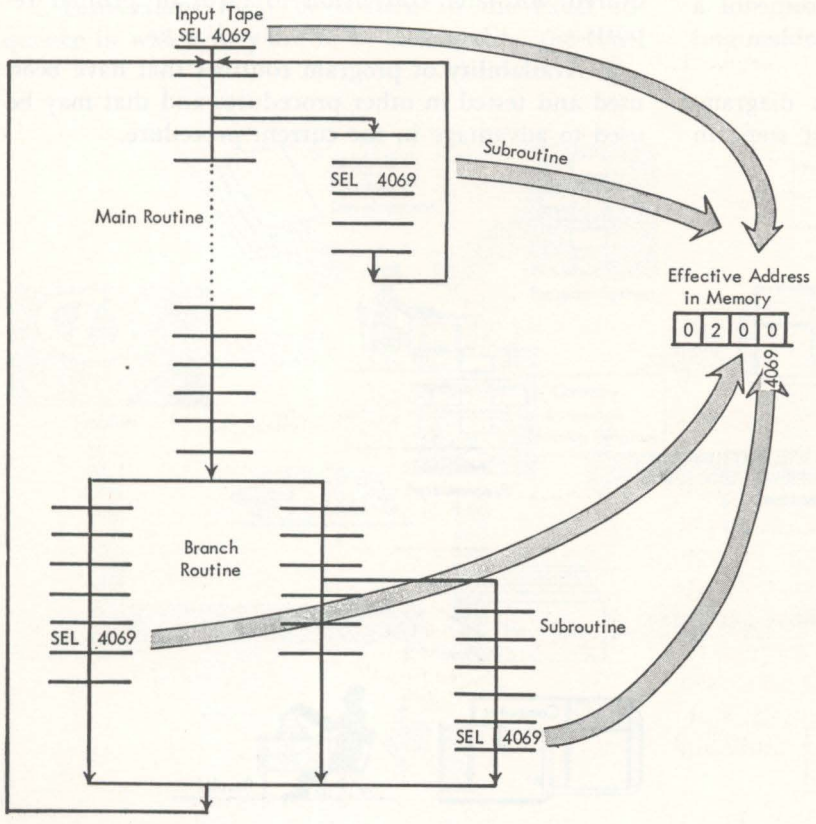


Figure 124. Indirect Address

Programming Systems

If the present pace of computer development continues, the over-all performance of data processing systems will soon be increased more than one hundred times. New systems will not only be much faster, but they will also be vastly improved in their ability to overlap and carry out a number of operations and procedures at the same time.

While the capability of computers is expanding at a fantastic rate, the technology of utilization and control is advancing at an equal pace. These improvements in techniques are as vitally important as the design of the data processing system itself. To a large extent, the future of computers depends not only on increases in speed, logical ability, and storage capacity, but also on the efficient utilization of these facilities as they are made available.

IBM programming systems have been developed to meet both present and future requirements of computer application.

Program Preparation

A computer program represents much more than a set of detailed instructions. It is the outcome of a programmer's applied knowledge of the problem and the operation of the computer system.

Problem definition, analysis, and block diagramming (see preceding section) are the first steps in program preparation. They are usually carried out independently of the computer and the programming system.

Some or all of the following must then be considered to prepare even the simplest program:

1. Allocation of storage locations to data, instructions, and related information.
2. Conversion of original data to an input medium.
3. Availability of reference data such as tables, files, or constant factors.
4. Requirements for accuracy and methods of checking and auditing.
5. Ability to restart the system in case of unscheduled interruptions or error conditions.
6. Automatic monitoring of the system to ascertain that the required input and output devices are connected and available for operation.
7. Housekeeping or set-up procedures that preset accumulators, switches, and registers; type operator messages; check file labels; and so on.
8. Format of output data with provisions, if required, for later conversion to cards or printed reports.
9. Availability of program routines that have been used and tested in other procedures and that may be used to advantage in the current procedure.

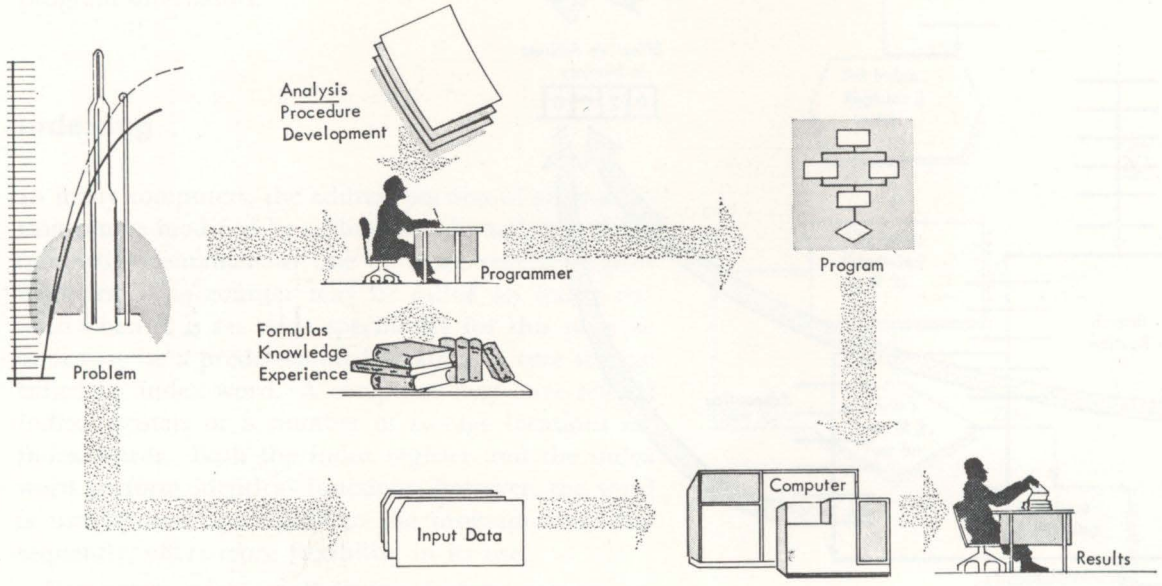


Figure 125. Direct Conversion of Problem to Machine Program

10. Conversion from the decimal number system to binary and from binary to decimal.

11. Editing of data with provision to record exceptions which cannot be processed.

Machine Coding

Figure 125 shows the basic relationship between the computer and programmer when the program is written in actual machine coding. The problem is first analyzed in terms of operations that the computer can perform. The program is then written in machine coding by the programmer who supplies tables, formulas, codes, or other reference material necessary for the specific application.

The problem then becomes input data, and the computer—by calculation or other operations—produces useful output.

A number of difficulties arise when the program is written in actual coding:

1. All instructions must be coded in machine language. With some computers, such as the IBM 704, 709, or 7090, which use binary representation in fixed words, this method of programming becomes impractical, if not impossible.

2. Instructions must be written in the exact sequence in which they are to be executed by the computer. If one or more instructions are omitted by error, all succeeding instructions must be relocated in storage to make room for insertions. This clerical accounting for all storage areas must be carried on entirely by the programmer.

3. The full burden of logic and program organization is placed directly on the programmer.

4. Previous experience—tested programs that might be utilized in part of the procedure—is difficult to work into the new program. Such programs must be linked to the new program by additional handwritten instructions.

5. The programmer must understand the computer in detail. He must know the location of each indicator or register, and he must program their functions entirely.

The Programming System

Many of the difficulties and inconveniences of writing programs directly in machine coding can be eliminated or simplified by the more advanced systems of program writing. Figure 126 shows the basic flow of work between problem and solution when a programming system is used.

The programming system consists of two parts: a language and a processor. The language is similar to

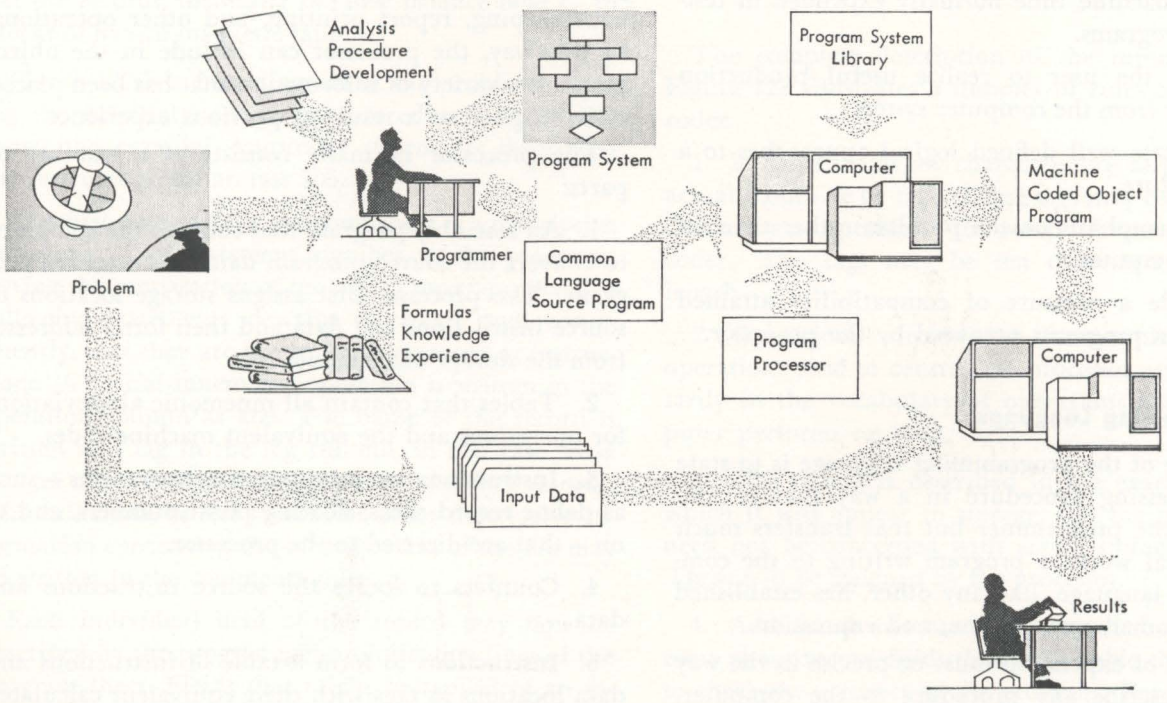


Figure 126. Conversion of Problem to Machine Program Using Programming System

the programmer's language and can be translated into machine language by the processor. The data processing procedure is first written in the programming language; this is called the source program. Then, the source program is translated into machine language, or the object program, by the processor.

When a programming system is used, the computer actually operates at two distinct levels:

1. As a translator or program assembly device.
2. As a data processing system.

At the first level, instructions in the programming language are translated into machine coded instructions. Storage areas are automatically assigned, constants or other reference factors are included, and library routines for checking, input-output, restart, housekeeping, and so on are assembled. Program routines may be generated from specifications furnished by the programmer. Normally, only one assembly process is necessary. The program thus produced may be used again and again to control the operation on data at the second level.

The programming system has several advantages. It can:

1. Save program preparation time.
2. Reduce clerical errors.
3. Simplify communication with the computer.
4. Utilize proved program techniques.
5. Use pre-checked routines.
6. Save machine time normally expended in testing programs.
7. Allow the user to realize useful production earlier from the computer system.
8. Facilitate well defined logical approaches to a procedure.
9. Place emphasis on the problem rather than on the computer.
10. Provide a measure of compatibility attained only in programs prepared by the processor.

The Programming Language

The purpose of the programming language is to state a data processing procedure in a way that is convenient for the programmer but that transfers much of the clerical work of program writing to the computer. This language, like any other, has established rules of grammar, punctuation, and expression.

The terms of expression must be precise in the way that they describe any procedure to the computer. These statements must convey to the computer exactly what it is to do, even though the aim of the language is to allow the programmer to state a procedure in a language nearly like his own.

The programming language must then compromise between the easiest way for the programmer to write, the design and organization of the computer system, and the requirements of the procedures to be expressed.

The language of the programming system may be either procedure-oriented or machine-oriented. If procedure-oriented, the language is independent of the computer and more closely approximates the everyday language of the user; it can be translated into several different machine languages using appropriate processors. If machine-oriented, the language is generally related to a specific data processing system and, therefore, lacks some of the compatibility inherent in procedure-oriented languages.

The Processor

The processor is a program that translates the programming language into machine coded instructions, make storage assignments, and assembles the instructions into a completed object program. This object program is then used by the computer to actually work the procedure.

Under direction of the programmer, the processor may draw on a library of information to assemble the machine coded program. This library can contain pretested routines for input-output, error checking, housekeeping, report printing, and other operations. In this way, the processor can include in the object program a variety of information that has been placed at its disposal as a result of previous experience.

The processor normally consists of a number of parts:

1. An assembly program that controls the computer to convert the source program data to a machine program. The processor first assigns storage locations to source instructions and data and then forms addresses from the storage assignments.

2. Tables that contain all mnemonic abbreviations for operations and the equivalent machine codes.

3. Instructions to interpret the operations — such as define record areas, floating point numbers, and so on — that are directed to the processor.

4. Counters to locate the source instructions and data.

5. Instructions to form a table of instructions and data locations as tags with their equivalent calculated storage locations. (The tag is a descriptive word or

phrase that labels a group of related data or instructions for the processor.)

6. Provision to edit the program and to produce messages to note errors. Editing functions include sequence checking of source instructions, all references to tags, and other miscellaneous checks that vary according to the design of the computer for which the source program is intended.

7. Provision to assemble the program for the system configuration of a specific computer.

Automatic Coding System

The following problem illustrates the basic programming method using the 705 Autocoder processor.

1. Read sequentially a file of 80-character warehouse inventory records punched in IBM cards. Each record includes two numeric fields A and B representing quantities of receipts and withdrawals. Field A is signed plus; field B, minus.

2. Add numeric fields A and B to produce a balance, C.

3. If C is plus, execute routine X; if C is minus, execute routine Y. (Routines X and Y are not described.)

4. After either routine X or Y is completed, write out the records, including the new balance field C, in blocks of five on magnetic tape.

Figure 127 is a flow diagram of the problem, showing the essential steps in the procedure. Figure 128 shows the essential Autocoder statements describing the instructions for an IBM 705.

The first entry on the program sheet is a descriptive statement. This statement signifies to the processor, during the preparation of the object program, that the following statements describe a record and, consequently, that they are not to be interpreted as instructions. A special mnemonic operation is written in the operation column as RCD. The name of the record is written as a tag in the tag column, in this case WHSE INV, an abbreviation for warehouse inventory records. The full name of the records or other descriptive information concerning the identification of the file may be written in the comments column.

Each individual field of the record may now be described by the programmer on following lines of the program sheet. Fields that might be typical of an inventory record are described as shown.

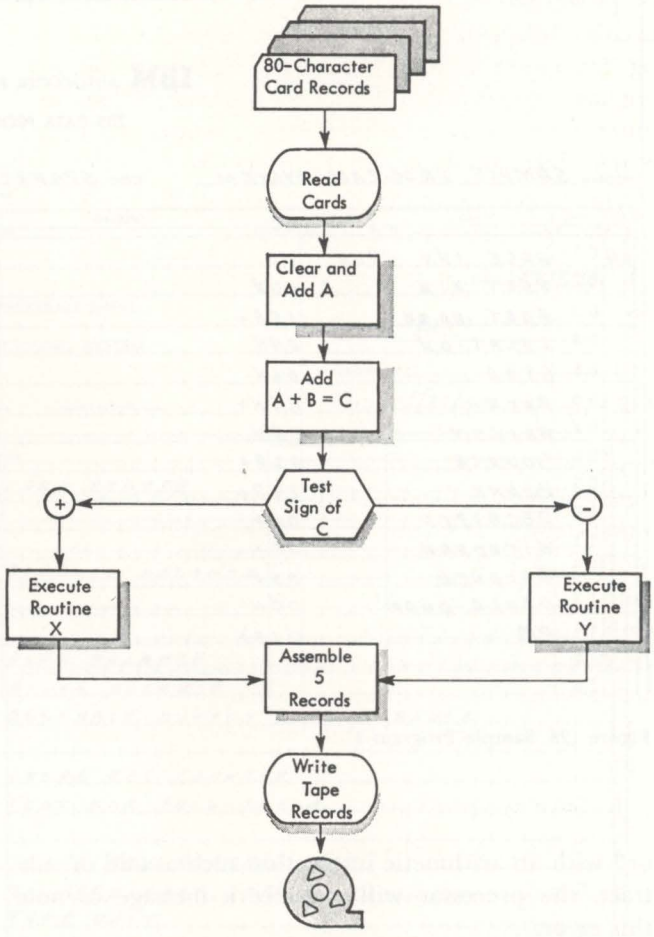


Figure 127. Problem to Illustrate Programming Methods

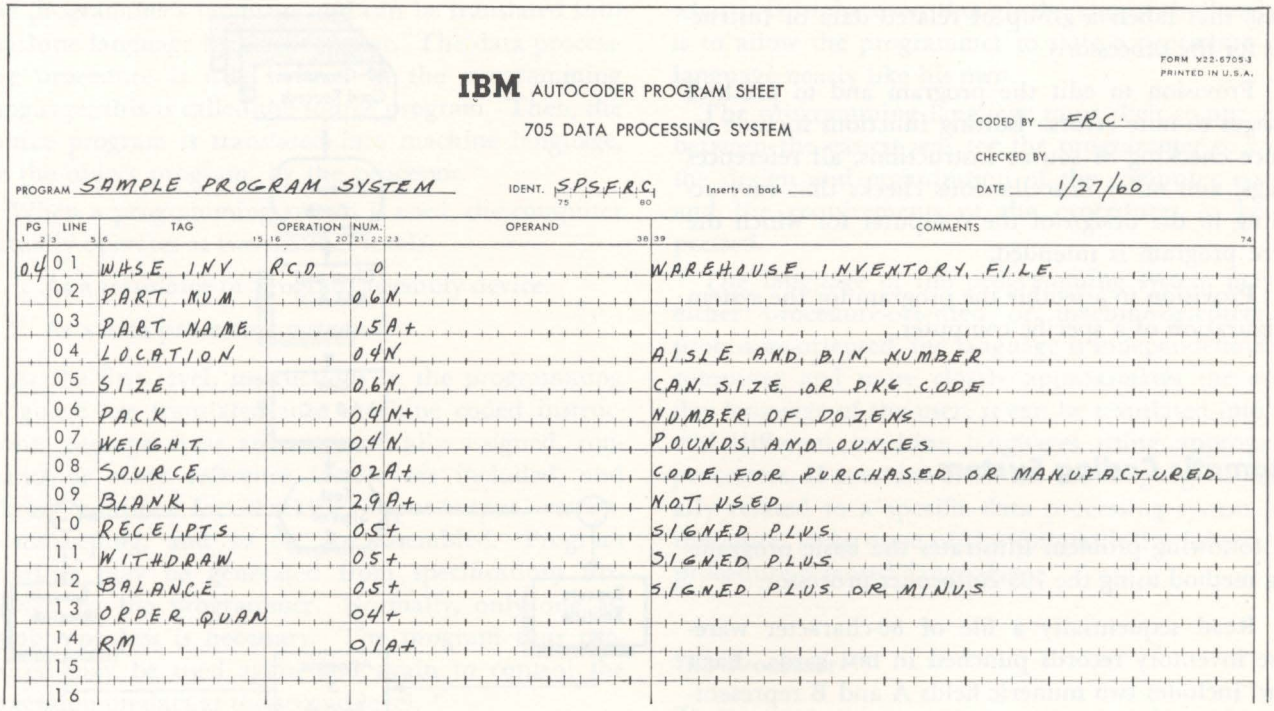
The complete description of the input record in Figure 128 illustrates a number of rules of the Autocoder.

1. Records, fields, or characters are called by their actual names or by tags. These tags may consist of any characters acceptable to the computer using the Autocoder. The tags may be ten characters or less in length.

2. Operations are written mnemonically; these operations used to control the processor are not necessarily in the vocabulary of operations that the computer performs on data.

3. The record is described in the exact format in which it will appear in storage, but the programmer need not be concerned with actual storage locations. These will be assigned by the processor.

4. Additional information is supplied to the processor about record fields that will enable the processor to check the accuracy of the program. For example, if the programmer refers to an unsigned field in the rec-



FORM X22-6705-3
PRINTED IN U.S.A.

IBM AUTOCODER PROGRAM SHEET

705 DATA PROCESSING SYSTEM

CODED BY FRC

CHECKED BY

PROGRAM SAMPLE PROGRAM SYSTEM IDENT. SPSFRC1 Inserts on back

DATE 1/27/60

PG	LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
04	01	WHSE,INV	RCD	0		WAREHOUSE INVENTORY FILE
	02	PART NUM		06N		
	03	PART NAME		15A+		
	04	LOCATION		04N		AISLE AND BIN NUMBER
	05	SIZE		06N		CAN SIZE OR PKG CODE
	06	PACK		04N+		NUMBER OF DOZENS
	07	WEIGHT		04N		POUNDS AND OUNCES
	08	SOURCE		02A+		CODE FOR PURCHASED OR MANUFACTURED
	09	BLANK		29A+		NOT USED
	10	RECEIPTS		05+		SIGNED PLUS
	11	WITHDRAW		05+		SIGNED PLUS
	12	BALANCE		05+		SIGNED PLUS OR MINUS
	13	ORDER QUAN		04+		
	14	RM		01A+		
	15					
	16					

Figure 128. Sample Program 1

ord with an arithmetic instruction such as add or subtract, the processor will produce a message to note this error.

5. The sequence of all entries is maintained by a page and line number preprinted on the program sheet. Additional space for comment to clarify the format of the record is provided as needed.

6. Other operations that describe the following entries to the processor are also included in the Autocoder system. These include CON to define a constant area, RPT to define a report area for printing, FPN for floating point number, and so on.

When the data areas are property defined, the writing of instructions is the next step in programming. Figure 129 shows Autocoder instructions for the main program.

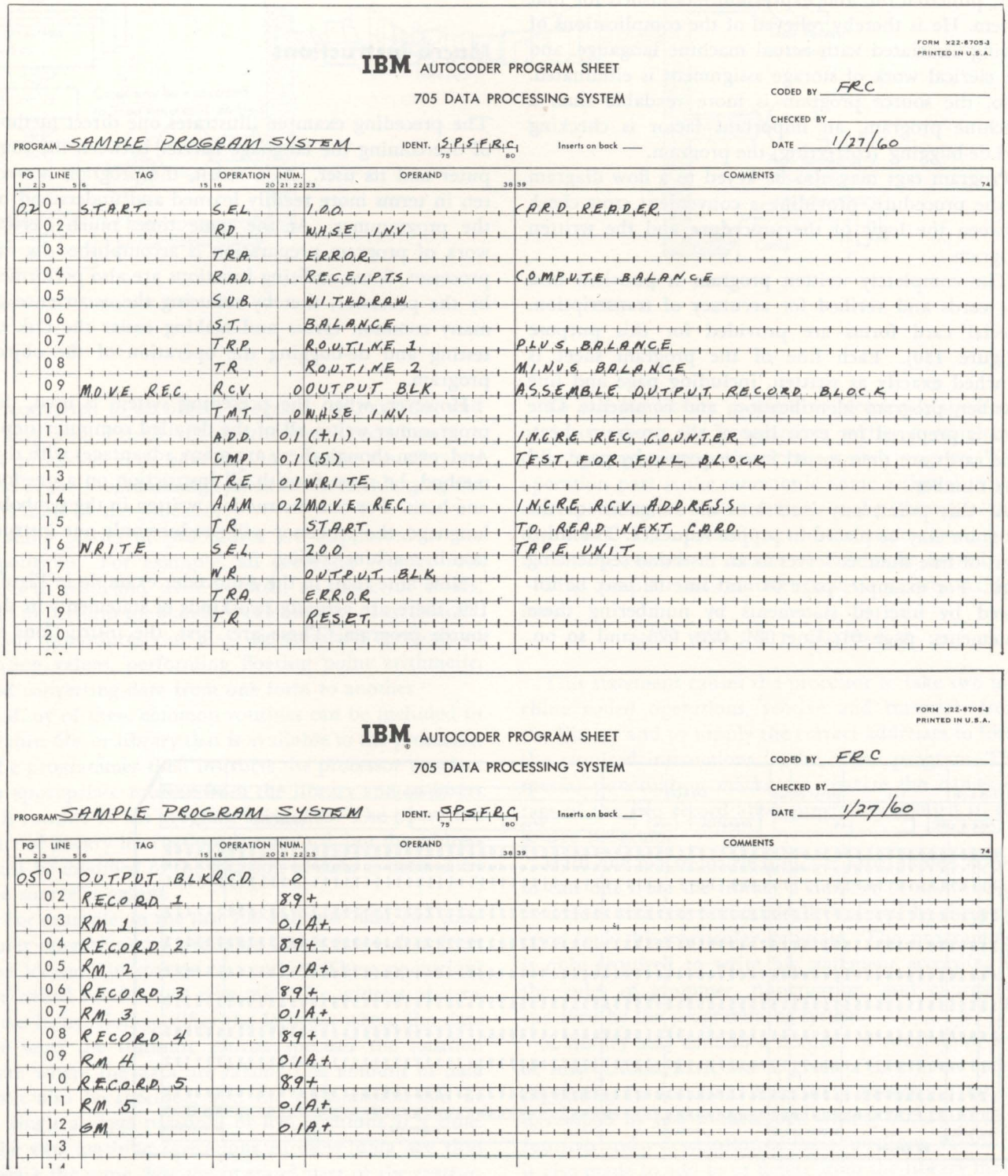
The operation part of each instruction is written as a mnemonic abbreviation. The processor interprets these standard abbreviations and substitutes the actual machine code for each.

The operand part of the instruction may be written in a number of forms. It may be the actual address of some device. For example, on line one, the operand 100 specifies a card reader. An operand may also refer to the tag of either data or instructions as shown on line 2. The operand is WHSE INV, and the complete statement RD WHSE INV means that the selected card reader is to read one record from the warehouse inventory file into storage. The actual location of any data does not have to be determined by the programmer but will be assigned by the processor.

The operand can also refer to the tag of other instructions in the program. For example, line 3 transfers to an error routine at location ERROR if either an end-of-file or an error condition occurs while reading a card. Also, lines 7 and 8 contain operands which are tags of the first instructions in routines 1 and 2.

An operand can also be a "literal." That is, the actual data to be operated upon may be written as the operand of the instruction. In Autocoder, the literal is designated by special punctuation: a parenthesis before and after the data as shown on line 11. The programmer adds the quantity one by surrounding a +1 with parenthesis in the operand column. The sum is placed in a specific storage register by designating 01 in the numeric column. Any of the 15 available storage units in the IBM 705 may be designated by writing its number in this column.

The resulting sum in the counter is compared against a 5 by the instruction in line 12. The digit 5 becomes literal data by proper punctuation. The purpose of the instructions on lines 11 and 12 is to count the number of records placed in the output block and to write these records on tape when five records have been assembled. When the number reaches five, the



IBM AUTOCODER PROGRAM SHEET

705 DATA PROCESSING SYSTEM

FORM X22-6705-3
PRINTED IN U.S.A.

CODED BY FRC

CHECKED BY

PROGRAM SAMPLE PROGRAM SYSTEM IDENT. SPSFRC Inserts on back

DATE 1/27/60

PG	LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
02	01	START	SEL		1,0,0	CARD READER
	02		RD		WHSE INV	
	03		TRA		ERROR	
	04		RAD		RECEIPTS	COMPUTE BALANCE
	05		SUB		WITHDRAW	
	06		ST		BALANCE	
	07		TRP		ROUTINE 1	PLUS BALANCE
	08		TR		ROUTINE 2	MINUS BALANCE
	09	MOVE REC	RCV		OOUTPUT BLK	ASSEMBLE OUTPUT RECORD BLOCK
	10		TMT		ONHSE INV	
	11		ADD	0,1	(+1)	INCRE REC COUNTER
	12		CMP	0,1	(5)	TEST FOR FULL BLOCK
	13		TRE	0,1	WRITE	
	14		AAM	0,2	MOVE REC	INCRE RCV ADDRESS
	15		TR		START	TO READ NEXT CARD
	16	WRITE	SEL		2,0,0	TAPE UNIT
	17		WR		OUTPUT BLK	
	18		TRA		ERROR	
	19		TR		RESET	
	20					

IBM AUTOCODER PROGRAM SHEET

705 DATA PROCESSING SYSTEM

FORM X22-6709-3
PRINTED IN U.S.A.

CODED BY FRC

CHECKED BY

PROGRAM SAMPLE PROGRAM SYSTEM IDENT. SPSFRC Inserts on back

DATE 1/27/60

PG	LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
05	01	OUTPUT BLK	RCD		0	
	02	RECORD 1			8,9+	
	03	RM 1			0,1A+	
	04	RECORD 2			8,9+	
	05	RM 2			0,1A+	
	06	RECORD 3			8,9+	
	07	RM 3			0,1A+	
	08	RECORD 4			8,9+	
	09	RM 4			0,1A+	
	10	RECORD 5			8,9+	
	11	RM 5			0,1A+	
	12	GM			0,1A+	
	13					
	14					

Figure 129. Sample Program 2

next instruction on line 13 transfers to the instruction at location WRITE. If the number is less than 5, a transfer is made back to the instruction at location START. Note that tags for instructions are required only for reference by other instructions.

To write instructions in any symbolic program system, the programmer merely follows the rules of grammer, punctuation, and expression established for that system. He is thereby relieved of the complications of coding associated with actual machine langauge, and the clerical work of storage assignment is eliminated. Also, the source program is more readable than a machine program, an important factor is checking and de-bugging (correcting) the program.

Program tags may also be keyed to a flow diagram of the procedure, providing a convenient cross check between the logic of the procedure and the written program.

The completely written program is punched into IBM cards and verified for accuracy of transcription. Special card forms are provided for this purpose (Figure 130). Each line of the program sheet is punched exactly as written, including page and line number, program identification, and comments. One card is prepared for each line of the program sheet. The cards are then sorted into sequence by page and line number.

At this point, any insertions or omissions to the program may be placed in proper sequence. The third digit of line number serves as an insertion sequencing digit. For example, page 01 and line 02 may be followed by inserted statements by numbering these statements: page 01; line 021, 022, 023, and so on. The system thus provides for the contingency of additions to the program without the necessity of relocating either instructions or data.

The source program cards become input to the processor and are acted on as data to produce an object program as output (Figure 131).

Macro-instructions

The preceding example illustrates one direct method of overcoming the language barrier between the computer and its user. As a result, the program is written in terms more readily learned and understood by the programmer. At the same time, much clerical work of program preparation is accomplished by the processor. Some auditing functions are also performed by the processor, thereby reducing the occurrence of many common errors and making easier the task of testing and de-bugging the operation of the object program.

However, when the preceding system is used, the programmer writes all of the detailed computer steps. And, even though there are many advantages with this method, he must furnish all instruction on a one-for-one basis. For one instruction written in the symbolic language, the processor will produce only one instruction in machine coding.

Now note that, in the Autocoder example in Figure 129, there are actually two kinds of statements in the source program. These are: first, the instructions to

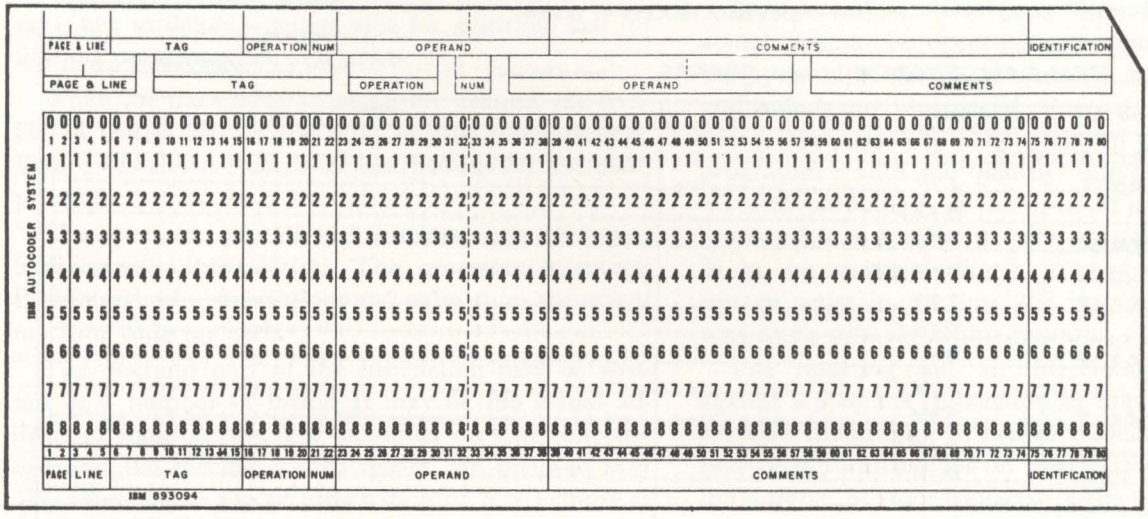


Figure 130. Autocoder Instruction Card

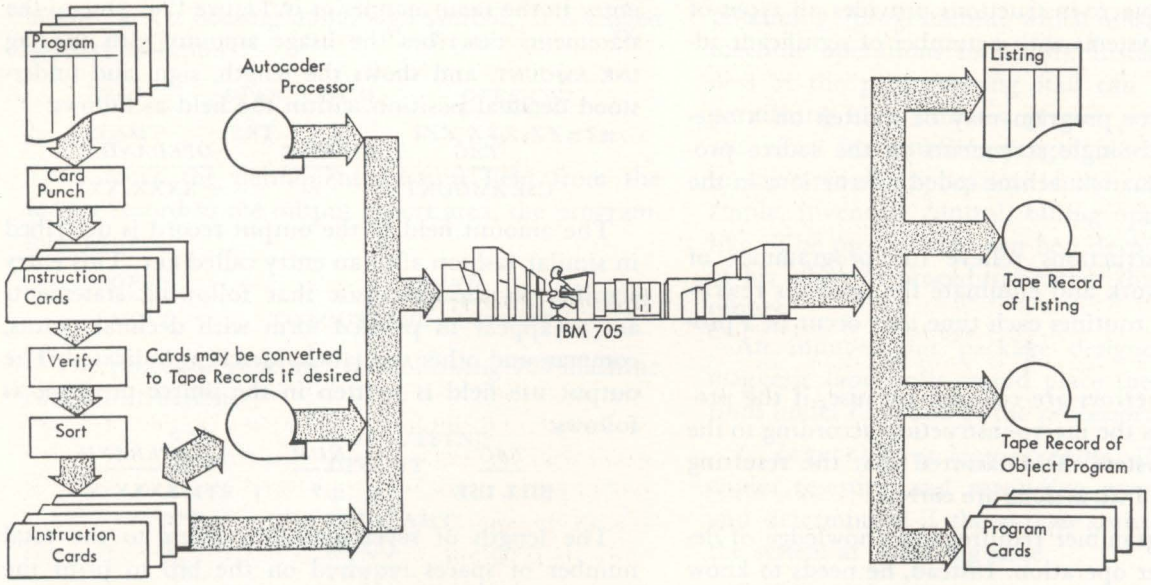


Figure 131. Autocoder Processor

be converted to an object program (SEL, RD, ADD, and so on); and, second, the instructions directed only to the processor. The next step to increase the effectiveness of the programming system involves enlarging the functions of the processor.

Many phases of procedures are duplicated or are repetitious. For example, all programs require routines to read and write records, to check for errors, and to perform the standard calculations of arithmetic. Many procedures can also use standard methods of scaling values, performing floating point arithmetic, and converting data from one form to another.

Many of these common routines can be included in a table, file, or library that is available to the processor. The programmer then instructs the processor to select an appropriate routine from the library and to insert this routine in his program as needed. The operation part of library instructions is standard, but the address must be modified to the location of data as defined by the source program.

For example, in the IBM 705 and IBM 7080, two computer operations are available which move data from one location in storage to another. These operations are called receive and transmit. The address of a receive instruction specifies the location where the data are sent; the transmit address specifies the location from where the data are taken. The amount of data must also be specified, and the transmission can be in single storage positions or five positions at a time. The two machine operations to move data are thus always the same, but the operand part of the instruction, produced by the processor, is variable.

To call for library routines or frequently used sets of instructions, the programmer writes a statement called a macro-instruction in the source program. The operation part is the mnemonic name assigned to the corresponding set of instructions in the library, in this case MOVE. The operand part of the macro-instruction names the records or blocks of data to be moved.

The complete macro-instruction is written as follows:

MOVE WHSE INV ¤ RECORD 1 ¤

This statement causes the processor to take two machine coded operations, receive and transmit, from the library and to supply the correct addresses to form the required instructions in the object program. The special punctuation marks ¤ separate the names or tags of the two record areas previously defined in the source program.

With one macro-instruction, the programmer is able to call out from the library a number of machine instructions that are automatically tailored by the processor to fit his particular program. The programmer is only required to write his statement according to the rules of grammer, punctuation, and expression specified by the programming system.

A library may contain a number of routines or sets of instructions. Each one is given a mnemonic name suggesting its function. Each data processing system developed by IBM is furnished with a library of such routines and a description of their functions. Provision is also made to add to or delete from the library material as required by the user.

The use of macro-instructions provides all types of programming systems with a number of significant advantages.

1. The source program may be written on a one-for-many basis. Single statements in the source program produce many machine coded instructions in the object program.

2. Macro-instructions relieve the programmer of some clerical work and eliminate the need for rewriting repetitious routines each time they occur in a program.

3. Program errors are reduced because, if the programmer writes the macro-instruction according to the rules of the system, he is assured that the resulting machine coded instructions are correct.

4. The programmer requires less knowledge of detailed computer operation. Instead, he needs to know the functions of the macro-instructions available to him. He is not necessarily required to completely understand the resulting machine coding. This is an important aspect of training and educational requirements for the programmer.

Program Compilers

So far, the functions of the programming system have been shown to involve mainly translation and assembly. A processor may also perform additional functions, specifically those of compiling the object program in the most efficient manner.

In the example shown in Figure 127, the operation is one of moving entire records from an input area to a grouped output area. However, when individual fields within a record area are to be moved, several other considerations may arise. Two examples follow:

1. A customer's bill for the consumption of product must show the amount used. This information enters the computer as a signed numeric field *understood* to contain two decimal places. This usage amount is to be printed in the customer's bill record with one decimal place. Therefore, the quantity is to be rounded to the nearest tenth of a unit. In this case, the data must be moved from the incoming record storage area to a register or accumulator where an arithmetic operation of rounding is done. The adjusted field is then stored in the output area in such a way that a decimal point appears between the units and tens position of the amount field for printing.

The programmer describes the input area by preceding a series of statements to that effect by an RCD entry in the same manner as in Figure 128. One of the statements describes the usage amount with the tag USE AMOUNT, and shows the length, sign, and understood decimal position within the field as follows:

TAG	*NUM*	*OPERAND*
USE AMOUNT	6	+ XXXX. XX

The amount field of the output record is described in similar fashion after an entry called RPT. This entry signifies to the processor that following statements are to appear in printed form with decimal points, commas, and other special characters as indicated. The output use field is written in the source program as follows:

TAG	*NUM*	*OPERAND*
BILL USE	7	XXXX. X

The length of seven positions refers to the total number of spaces required on the bill to print the usage amount. The position of the decimal point is shown in the operand column.

To prepare the amount field for printing, the programmer writes the macro-instruction MOVE as:

OPER	*OPERAND*
MOVE	USE AMOUNT ¤ BILL USE ¤

From the information given the processor, the following machine coded instructions are produced:

RAD	USE AMOUNT
RND	1
SPR	BILL USE

In the IBM 705 or 7080, these instructions reset and add the signed use field into the accumulator, round the amount one position, and place the amount in the output area for printing with a decimal point character in the proper space. From the information supplied the processor in the source program, the instructions that carry out the programmer's intent are compiled.

2. In producing a report of delinquent accounts, it is required to print the total cash amount outstanding. This amount has been computed and stored in a master record pertaining to the account in question. It is a seven-digit signed field representing a cash amount in tens of thousands of dollars. The delinquency report is to be printed in dollars and cents with a floating dollar sign. A comma should appear if the amount exceeds $999.99. The field as it appears in the master record is tagged TOTOUTSTND and is defined as an RCD, as follows:

TAG	*NUM*	*OPERAND*
TOTOUTSTND	7	+ XXXXX. XX

The output format is defined as an RPT with two decimal places, a comma following the thousands po-

sitions, and a floating dollar sign symbol. It is tagged DELAMT as follows:

TAG	*OPER*	*NUM*	*OPERAND*
DELAMT	RPT	11	$XX,XXX.XX¤$¤

To move the delinquent amount field from the master record to the output report area, the programmer writes:

OPER	*OPERAND*
MOVE	TOTOUTSTND¤DELAMT¤

The processor generates the following 705 machine coded instructions:

```
RAD      TOTOUTSTND
RCVS     DELAMT
TMTS  1  #, #
SPR      L, DELAMT
RCVS     PRTST
TSL      PRTST
```

In addition, certain constant factors are supplied to place the dollar sign. Because the above instructions are peculiar to the 705, the detailed operation involved is not explained. They are shown merely as an example of the enlarged functions of a processor in producing a series of complete, detailed machine instructions from a single macro-instruction. In the 705, the object progam would:

a. Place a comma in the proper storage position for printing.

b. Transfer the data from the master record to the delinquency report, suppressing leading zeros and deleting the comma if the amount is less than $999.99.

c. Place the dollar sign adjacent to the first significant digit of the report field.

Program Package

A number of other developments advance the concept of placing additional functions of detailed program writing with the system processor. One of these developments is the input-output package.

The package is a series of program routines available to the processor. When these routines are included in the object program, they handle all operations of some type of input-output device. The programmer calls for these routines by writing descriptive statements in the source program. The language of these statements then becomes a portion of the complete language of the programming system.

The use of a package eliminates much time and effort formerly required for machine-oriented programming (programming which must consider specific machine operations in detail). Instead, the effectiveness of the programming staff can be improved by permitting it to utilize a greater portion of its time and effort for installing programs which are oriented to the over-all application of the computer – for example, inventory control, billing operations, and the like. The programmer can best devote the major part of his effort to procedural rather than machine techniques.

An input-output package designed for handling magnetic tape units would place the following functions under direction of the processor:

1. Input-Output Housekeeping. This function includes resetting and rewinding magnetic tape units and determining if the proper units are attached to the system at operating time.

2. Label Handling. Normally, magnetic tape files have standard labels which precede the file data on each reel of tape. Labels are checked during the operating procedure to determine if files called for by the program are on the assigned tape units. Labels also include provision for dating the files, counting the number of records, and other controls that establish efficiency in handling tape records.

3. Scheduling Data Flow. When multiple input and output units are used, this function may overlap the operations of reading and writing with computing. The flow of data into and out of the system must be scheduled for optimum efficiency.

4. Execution of Input-Output Operation. These are the actual operations of reading and writing.

5. Error Detection and Correction. The package includes standard routines for checking all possible error conditions and correcting them, if possible.

6. End-of-file Procedures. These procedures provide for rewinding tapes, typing messages to indicate the completion of a job, reel changes, recording controls, and other information.

7. Record the Contents of Main Storage and Restart. These functions include procedures that allow the system to restart with a minimum of lost time if an error or other unplanned condition arises. The procedure can also be checked at predetermined intervals.

8. Blocking and Deblocking of Tape Records. Many magnetic tape files are written with several records grouped between inter-record gaps. Provision may be included in the input-output package to facilitate the handling of such grouped records and to reassemble the records into some other specified output grouping as required.

The package can also include routines that control input-output devices other than magnetic tape units. The features of the package would be similar to the example.

Use of these common routines encourages the establishment of programming and operating standards. The advantages of standard methods are significant, especially because they allow programmers to communicate more easily with each other. This is particularly important during testing phases of programs when testing is carried out by persons other than those who have written the source program. Standard methods enable the machine operator to become familiar with conditions that mean the same in every program.

Other packages accomplish additional procedural functions. These include report writing, file maintenance, decision-making, and miscellaneous computing routines.

The statements in the source program which call for the packaged routines are written in the language of the programming system. The action of the processor is broadened and enlarged to translate this language. The processor now does much more than translate. It determines the intent of the programmer and assembles, compiles, and generates the necessary instructions to complete the object program.

The organization of the programming system with packaged routines is shown in Figure 132. The interpretation and adaptation of the packaged routines normally takes place in a separate stage of the processing run. The routines are first translated by the processor to one-for-one symbolic statements and are then assembled to machine coded instructions. Other intermediate stages of preparation may be carried out by the processor; for example, the packaged routines may include macro-instructions which in turn are made up of detailed instructions either in machine coded or symbolic form.

Fortran

Programming systems now in advanced stages of development and operation almost completely remove the programmer from any consideration of the actual characteristics of the computer, except for a general knowledge of machine operating principles and the types of input and output that can be handled. One outstanding system, produced through the combined efforts of IBM and its customers, is Fortran (Mathematical *For*mula *Tran*slation System).

Source programs in this system are written in the Fortran language, which closely resembles the ordinary language of mathematics. Processors have been developed to produce object programs for the IBM 650, 704, 709, 705, 7080, and 7070 Data Processing Systems.

The Fortran language is intended to allow expression of any problem of numerical computation. In particular, problems containing large sets of formulas and many variables can be dealt with easily. The language of Fortran may be expanded by the use of subprograms in much the same manner as previously explained for packaged routines. These subprograms may also be written in Fortran language, and may be

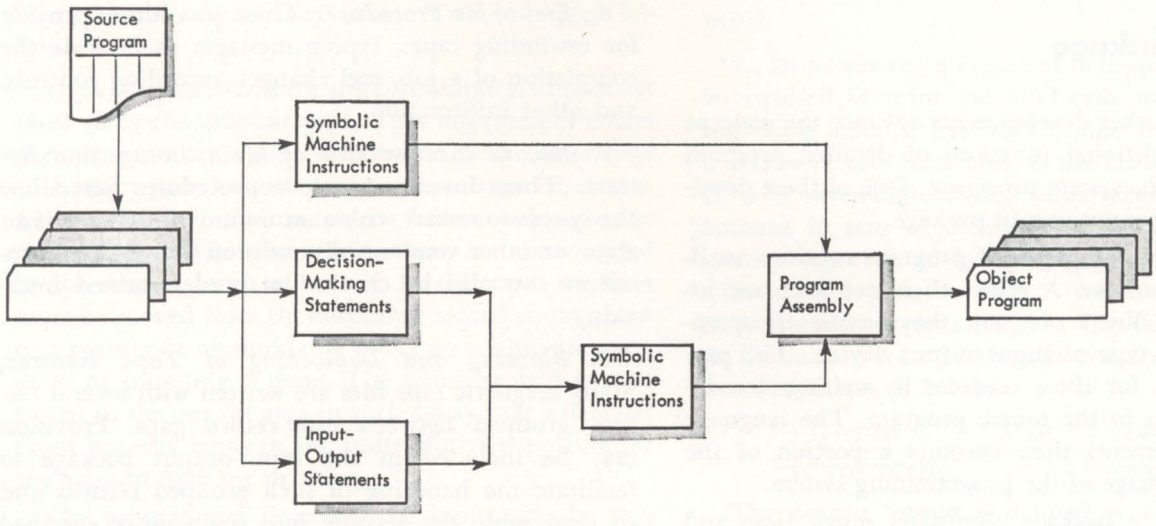


Figure 132. Stages of Program Conversion with Packaged Routines

called for by the other Fortran programs and subprograms. Thus, the language may be expanded by this use of subprograms to any desired depth.

Because Fortran is primarily intended for problems which have a numeric meaning, the language is less satisfactory for the commercial-type problem. Nevertheless, many logical operations which cannot be directly expressed in Fortran can be carried out by incorporating suitable library routines for this purpose.

Figure 133 is a flow diagram of a part of a typical invoice preparation problem, including computation of gross amount, net amount, and discount and the writing of each item on a separate line. Figure 134 shows the same problem written as Fortran statements.

This system of language represents an important step in the development of universal languages which may be easily understood by both man and machine.

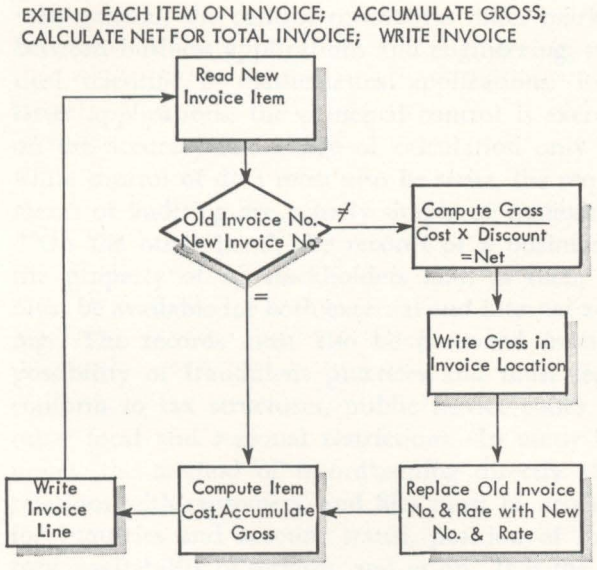


Figure 133. Invoice Preparation

```
10   READ 1, INVNEW, DSTNEW, IQUANT, UNICST
     IF (INVPRE) 15, 30, 15
15   IF (INVNEW-INVPRE) 20, 30, 20
20   CSTNEW = DSTPRE * GRSCST
     PRINT 2
     PRINT 3, INVPRE, GRSCST, DSTPRE, CSTNET
     GRSCST = 0.
30   QUANT = IQUANT
     GRSCST = GRSCST - QUANT * UNICST
     INVPRE = INVNEW
     DSTPRE = DST NEW
     GO TO 10
1    FORMAT (18, F15.2, 18, F15.2)
2    FORMAT (45H b PREV. bINV.bbGROSSbCOSTbbPREV.bDSCbbNETbCOST)
3    FORMAT (18, F13.2, F10.2, F13.2)
```

Figure 134. Fortran Statement

Sort Programs

In addition to the common functions that appear within many different procedures, there are also certain complete procedures which must be carried out in all data processing installations. These may be standardized by type of machine system and basic application.

For example, the operation of sorting tape files occupies a large percentage of computer time in any application where records are processed sequentially. Incoming records must be arranged to conform to the order of other existing files with which they are to be merged or compared. Output data may be written in one sequence to produce a required tabulation or report, but must be sorted in another sequence for other reports or for further processing.

Because the primary input and output of any computer is magnetic tape, a number of generalized programs have been developed by IBM to sort tape records. Such a program is generalized in that it is capable of modifying itself according to the given specifications of the records to be sorted. The program can thus perform a large variety of sort applications on the system for which it is designed.

A typical sort program accepts either single or blocked fixed-length records or single variable-length records. The sorted output may also be either single or blocked fixed-length records or single variable-length records. The characteristics of records in the file are usually specified by the operator on a punched control card. These characteristics include length and location of control data fields within the record, record length, and input and output blocking. The card can also specify characteristics of the sorting procedure itself, such as input and output units to be used, provision for file labels, and internal storage available to the sort.

Generalized sort programs also include provision for check point, restart, and interruption procedures. Hash totals, record counts, and sequence checks are provided to verify the accuracy of processing.

These programs are designed to achieve maximum system efficiency, using simultaneous reading and writing whenever possible and the best internal record grouping according to the available capacity of storage.

Utility Programs

Utility programs involve certain functions common to all data processing applications. These functions include the program controlled operations that involve loading programs into storage, tracing errors, recording the contents of storage, and so on.

Standard loading programs, for example, have been written for all types of computers. The program card precedes the operating program and arranges the instructions in storage in the sequence specified by the programmer. The load program is usually punched in one or more cards at the beginning of the program deck or it may precede the operating program on tape. The load program is placed in specified areas of storage by manual instructions from the console or by depressing a load key. The machine is reset to the first instruction and automatic operation begins. This operation remains under control of the load program.

Instructions of the operating program are read and placed in storage. Operation continues until the complete operating program and constants or other data are loaded.

When the operating program is completely loaded, control of the machine is automatically transferred from the load program, and reading of input data can begin. Automatic operation continues until all data are processed or operation is halted by instruction.

If the processing operation is interrupted it may be necessary to record the contents of storage. A storage print-out utility program can be used for this purpose. This recording ability is particularly desirable in a testing operation when the machine is halted by a program error. Analysis of the printed contents of storage is useful in determining the cause of error.

Other utility programs developed by IBM and its customers are available as an aid to system operation.

A data processing procedure must include two main areas of activity: accomplishment of the desired result and control of the procedure itself. Complete controls are far broader than the checks designed to supervise the quality of work produced by the computer system. These controls must also consider the entire application and its importance in a business or scientific endeavor.

Methods must be devised to control the flow of information into and out of the data processing system and to assure that all information received is correctly included in the required results. In addition, should omission or duplication occur, methods must be devised to establish an audit trail without completely retracing an entire procedure.

In general, the control procedures differ markedly between business applications and engineering, statistical, scientific, or mathematical applications. In the latter applications, the principal control is exercised on the accuracy and range of calculation only and, while control of data must also be strict, the requirements of auditing are usually simple or nonexistent.

On the other hand, the records of a business are the property of its stockholders and, as such, they must be available for both external and internal auditing. The records must also be protected from the possibility of fraudulent practices and must legally conform to tax structures, public service codes, and other local and regional restrictions. In many businesses, the method of recordkeeping directly affects relations with customers, and files must be accessible for inquiries and account status, position of inventory, availability of services, and so on. It is the area of business practices with which the following discussion is primarily concerned.

In any application, however, the purposes of the over-all procedure control are to:

1. Assure that data entering the computer are accurate.
2. Check clerical handling of data before they reach the computer to assure that the data are complete and not duplicated.
3. Arrange data in the form best suited for economical use by the computer.
4. Provide a means of auditing the steps of the complete procedure so that, in the event of error or inconsistency, the trouble may be located with minimum lost time.
5. Assure that accounting for tax purposes or other legal requirements is carried out according to law.
6. Guard against fraudulent practices that may affect the financial standing or reputation of the business.

Control Groups

The data processing center is usually a service unit. It receives information to be processed, perhaps in the form of punched cards, from outside sources, makes the necessary calculations, and produces the necessary reports. The center, as a rule, originates no information but is concerned only with data sent to it. Under these circumstances, it is possible to establish controls over the employees of the center and to prevent fraudulent or inaccurate handling. These controls are usually delegated to some group, either organized within the center or closely associated with it for this purpose.

Control of Payroll Data

One large company computing its payroll with an electronic installation established an office known as the payroll bureau. All changes in payroll data — such as rate changes, new employees hired, terminations, and changes in 25 types of pay deductions — are routed through this bureau. The changes are indicated on an authorization form, prepared in duplicate in the originating department.

The original approved copy of this document is forwarded to the payroll bureau. In the bureau, the change authorization forms are grouped by type of pay data affected. Adding machine totals of numeric fields are accumulated, regardless of whether these fields represent dollar information or identification; in the case of new employees or terminations, the number of employees affected is also included in these control totals. Cards are then punched and key verified for input to the computer.

The totals of the changes are accumulated weekly from the cards on a conventional punched card accounting machine. These totals are checked with the adding machine totals previously prepared. During one of the computer runs, normal pay and pay deductions are calculated. This sum is sent to the payroll bureau where the totals are compared with those previously recorded. This control over payroll

changes not only incidentally checks the operation of the data processing system but also checks the clerical handling and accumulation of information as it enters the system.

Control of Sales Data

Procedures can be controlled to assure the best relations with customers. One example of this control is supervision of the company's price structure for the commodities or services it sells. Here, a control group might be established to compute amounts to be charged to a customer.

In many organizations, for example, the sales department has discretion over the prices that customers are charged. However, deviations from established prices may occur because of allowances for defective merchandise or because of special situations. The function of the control group is to see that all exceptions from the official price list are investigated and can be explained. The computer is used to report these exceptions.

The identification code of merchandise shipped and the code number of the customers are entered into the computer, along with special notification if the sales price differs from that shown in the master price file. The computer prices all shipments, except those for which it has special notification, at the master price rate and follows special instructions for the exceptions. When the output tape that will eventually print the invoices is produced, a special listing is made of all shipments that have been calculated at other than master file prices. The listing is forwarded to the control group for investigation.

The control group may have other control functions that relate to the contents of the master files. An electronic system differs significantly from a manual system in the method of referring to the authorized selling price information. Where invoices are computed by billing clerks, it is reasonable to assume that, while individual clerical errors may occur, original price information is accurate. This is so because, as a rule, the clerks use an identical, up-to-date price list.

In the computer system, on the other hand, the prices are inserted from a master price tape and reference is made to this tape by the machine in determining the billing price. Therefore, if an error is made in the price of a particular product as recorded on the master tape, this error is reflected in the billings for all customers purchasing this item. Therefore, the contents of this master file must be strictly controlled.

Changes to the file may be made during a special run by the computer. Price changes are inserted and a change register is produced. A copy of the change register is forwarded to the control group where it can be examined in detail. From time to time, the master tape can be used to prepare a complete printout of portions or all of the product master files. It may also be used, if descriptive information is present, to prepare a price catalog.

Similar controls can be established to make certain that shipment is made only to customers of acceptable credit standing. This control can be established by a customer master file that is used to extract the proper shipping and billing addresses and to carry the approved credit limits. By proper control, it can be determined that shipments are made only to customers whose credit standing is approved by the credit department. Companies doing business with franchised dealers can use this procedure to detect shipments to unauthorized dealers, because the absence of a name in the customer master file immediately prevents invoice preparation.

Systems Checks

Systems checks are designed to control the over-all operation of a procedure within the computer system. They insure that all required data are received for processing and that all data leaving the system are complete and accurate.

Systems checks may include controls to insure that all input records are in file for a current processing period and that incorrect or unrelated records are excluded. These checks may also verify the distribution of detail transactions to update records when such distribution is made by coding. Systems checks may be devised for factors developed during calculation to compare this logical or reasonable relationship with other known factors. Crossfooting totals is a commonly used systems check to prove calculation or accumulation of quantities and values.

The types of of systems checks vary with each application of the computer and with the kind of equipment used. Particular attention should be paid to including systems checks during the early stages of application planning, because such controls can be most effectively fitted into the program at that time. It is sometimes advisable to modify the procedure to include the most efficient controls; this is usually far less costly than designing a procedure without required controls.

Many commercial procedures require strict accounting control with provision for audit trails. This required control means that the program must be designed to take full advantage of the high reliability

built into the data processing system. To meet the requirements of efficient operation of the entire system, this control also means that the trouble can be localized quickly in case of error, without retracing the entire procedure or reprocessing all records.

In some machines, built-in checking features make detailed systems checks unnecessary. In others, data manipulation in the central processing unit is less stringently checked, particularly where elaborate checking circuitry would materially increase the cost of the system without a proportionate increase in accuracy.

Checking is a form of quality control. It follows that, when some percentage of error can be tolerated, checks may be used more sparingly. Some typical systems checks follow.

Record Count

A record count is a tally of the number of records in a file. The count is normally established when the file is assembled. In the case of magnetic tape, this count would be established when the file is written.

The total number of records is carried as a control total at the end of the file or reel and is adjusted whenever records are added or deleted. Each time the file is processed, the records are re-counted, and the quantity is balanced against the original or adjusted total. If the re-count agrees with the control total, it is accepted as proof that all records have been run.

Record counts may also be established by batches. This is desirable when source data are to be put into the procedure for the first time.

Although the record count is useful as a proof of processing, it is difficult to determine the cause of error if the controls are out of balance. A failure to balance does not help to locate a missing record nor does it indicate which record has been processed more than once. Therefore, some provision must be made to check the file against the source records, a duplicate file, or a listing known to contain the proper number of records.

An incorrect record count usually indicates a machine failure when tape records are being processed because, once written on the tape correctly, records cannot be misplaced or lost. In this case, the doubtful portion of the file should be rerun for correction.

Control Total

The control total may be made up from amount or quantity fields in a group of records. It is accumulated manually or by machine when the file is originated or when a quantity is first calculated. The control total can be either a grand total or more convenient intermediate or minor totals.

When the file or group of records is processed, the fields are again accumulated and balanced against the control total. If the total is in balance, it serves as proof of processing all records correctly.

The control total is an efficient systems check when it can be used to predetermine the results of calculation or the updating of some record. For example, when preparing to process a payroll, the total number of hours worked by all employees is pre-established from clock or job-card records. This figure then becomes the control total for payroll hours for all subsequent reports. Totals may be broken down by group or department. The sum of all totals must balance back to the complete original total.

Control totals are normally established for batches of convenient size, such as department, location, account, or division. By this method, each group of records may be balanced as it is processed. Corrective action, if needed, is limited to small, easily checked groups rather than to one grand total.

Proof Figures

Proof figures may be used to check an important multiplication in a procedure. As such a check, the proof figure becomes both a systems check and a check on the operation of the computer. The proof figure is usually additional information carried in the record.

An example of the proof figure is the multiplication of quantity by unit cost. The check is based on the relationship between actual unit cost and a so-called proof cost. An arbitrary fixed figure (Z) larger than any normal cost is set up. (If a cost range for all products in a given file is from \$.50 to \$10.95, Z might equal \$11.00.) Proof cost is the remainder when cost is subtracted from Z, or proof cost may be expressed by the formula:

$$\text{Cost} + \text{Proof Cost} = Z$$

Proof cost is carried as an extra factor in each record. Z is a constant which can be placed in storage for use when the proof figure is calculated.

Whenever quantity is multiplied by cost, it is also multiplied by proof cost. Normally, the factors accumulated during processing are quantity, quantity $\times$ cost, and quantity $\times$ proof cost. At any point, it is possible to check the sums (Σ) of all factors accumulated up to this point as follows:

$$\Sigma(\text{Qty} \times \text{Cost}) + \Sigma(\text{Qty} \times \text{Proof Cost}) = \Sigma(\text{Qty} \times Z)$$

The left side of the equation can be calculated by a single addition of the two progressive totals that

have been accumulated during the procedure. The right side of the equation can be calculated by a multiplication of the accumulated quantity and the constant factor Z. This check insures that each particular multiplication was performed correctly.

This type of proof figure can be applied to other applications by using the same general approach.

Limit Check

A limit check is a test of record fields or programmed totals to establish whether certain predetermined limits have been exceeded. For example, if transaction codes for certain records are known to cover only the digits 0 through 5, a check can be programmed to see that no code exceeds the limit 5.

A second type of limit check assures that calculated totals are reasonable. Some quantities or values in a procedure never vary more than a given percentage between processing periods.

Payroll procedures often contain many limiting factors that can be checked by the program. The upper limit of gross pay is usually determined by the type of payroll: hourly, salary, piece rate, incentive, and so on. Hourly rates must fall within established wage scales. The total number of hours worked per employee is also subject to certain limits.

Limit checks may also be used in table look-up procedures. If an item is known to be in a given table in storage, the modified table address may be checked against the address of the upper table limit to verify correctness of the search. If the search begins to exceed the limits of the table, an error has occurred and corrective action is required.

In many mathematical problems, the range of the final calculation can generally be estimated. If a result falls outside this reasonable range, it may be assumed that some error condition is present, either in the data, in the program, or in the calculation. Departures from normal trends may also indicate faulty procedures. The simple application of a limit check in such problems may save much detailed checking, with consequent simplification of the program.

Crossfooting Checks

Crossfooting checks may be used to check known control totals, or they may serve as proof totals originated during a procedure. For example, during the processing of employee records in a payroll, calculations develop amounts of gross pay, taxes, deductions, and net pay. Normally these amounts are accumulated by department or other convenient batch controls. The totals of gross pay at any point should be equal to the totals of net pay, deductions, and taxes.

Tape Label

File identification recorded at the beginning of a reel of magnetic tape is called a tape label. The label may specify the job total and/or number, date of last processing, number of the reel, and so on. A label may also be placed at the end of the file or reel.

The labels are read into storage at the beginning and end of the program as an added control that the proper records have been processed. The label may also insure a true end-of-file or end-of-job and may also include a record count.

Housekeeping Checks

The first instructions of nearly every program are intended to perform functions of housekeeping in preparation for processing. These instructions may set program switches, clear accumulators or registers, set up print areas, move constants, and so on. In addition, housekeeping instructions may perform systems checks by testing to determine if all input-output units required by the main program are attached to the system and ready for operation. File labels may be checked and up-dated, constant factors may be calculated, and other information pertinent to the proper operation of the system may be called to the operator's attention by programmed instructions.

Check Point and Restart

A check point procedure is a programmed checking routine performed at specific processing intervals or check points. Its purpose is to determine that processing has been performed correctly up to some designated point. If processing is correct, the status of the machine is recorded, usually by writing this information on a tape. The normal procedure is then continued until the next check point is reached.

Check point procedures have the effect of breaking up a long job into a series of small ones. Each portion of the work is run as a separate and independent part and each part is checked after it is completed. If the check is correct, enough information is written out to make it possible to return to this last point automatically. If not, the portion of work just completed incorrectly is discarded and the system restarts from the last point at which the work is known to be correct. A restart procedure:

1. Backs up the entire computer system to the specified point in the procedure, usually to a check

point. Tape files are backspaced or rewound; card units and printers must be adjusted manually.

2. Restores the storage of the computer to its status at the preceding check point. This may include the adjustment of accumulated totals, reloading the program itself, re-establishing switches and counters, restoring constant factors and so on.

The proper use of check point and restart procedures in a program contributes to over-all operating efficiency of a computer system. In the event of power failure or serious machine malfunction, they provide a means of rerunning only a small part of a job without having to rework an entire job. This may mean a saving of many hours of machine time.

Restart procedures also allow interruption of a given job for the scheduling of other jobs that need immediate or emergency attention. Thus, any procedure may be interrupted intentionally by the operator and replaced with another job when necessary. Provision for restart is also convenient at the end of a shift or other work period when the operation of a job must be terminated without loss of production time. Restart procedures also provide interruption of machine operation for emergency repairs or unscheduled maintenance.

Machine Checking

Procedures perform two functions. First, they accomplish useful work; second, they control quality and accuracy of the work.

In the data processing procedure, useful work consists of operations such as sorting, calculating, collating, reading, and printing. Control operations are necessary to establish and maintain accounting controls, calculation checks, and machine checks. The programmer can use these checking devices at his own discretion. Basically, two types of checks may be written:

1. Checks on the validity of data handled by the input-output units.

2. Checks on the handling of data within the computer, including checking for legitimate instruction code, arithmetic overflow, valid signs of numeric quantities, and other check indicators.

In many cases, it is not necessary to interrupt machine operation (to halt the computer) when an error condition is detected. The programmer may insert special transfer or branch instructions designed to handle certain types of errors as exceptions. An error in reading a record from tape, for example, may be programmed to backspace the tape and reread the record. If a correct reading is obtained the second time, normal operation continues. If the error persists, operation can be interrupted or the incorrect record can be noted and operation continued.

In some systems, the error indicator initiates a special routine that places the computer under control of a repair subroutine, accomplishes the program repair, and then proceeds back to the main program for continued processing. This operation is completely automatic.

In other systems, appropriate testing instructions accomplish the same end result. In all systems, however, the interrogation or interruption is under the programmer's control and discretion.

Some machine check indicators, however, stop all processing immediately. This type of indicator includes such conditions as: a blown fuse in the computer, air or humidity conditions exceeding prescribed limits, broken magnetic tape, or a card jam in card equipment. All of these cases must be brought to the operator's attention immediately and, therefore, they cause the computer to halt.

Appendix

Business Practices

Business needs a tremendous amount and variety of information to operate effectively. The business with the best communications system has a distinct advantage over its competitors. As the pace of business quickens, it is imperative that time lags between an event and its appearance in a useful report be minimized. In addition to reporting speed, the arrangement of information must permit rapid pinpointing of areas requiring attention.

Reporting past events, however, is not enough. New techniques, which permit data to be used more and more as a management tool for policy planning, are gaining importance as business competition intensifies.

This section outlines areas of business where data processing systems have been profitably used.

Within each category of business (Figure A), it is difficult to find two organizations doing business exactly the same way. Because a business is an organization of people, perfect coincidence of methods of operation between two companies is improbable.

Each business has terms of the trade which mean nothing to another. For example, a procedure description in the textile industry reads: "The daily warp status report is prepared on the accounting machine showing the number of looms mounted, warps drawn, warps drawing, warp not drawn (spread by top and bottom beams) and number of warps slashing."

Examples of this kind of jargon are found in any of the 15 business categories listed. But, while the differences are endless, the similarities are also numerous — and much more meaningful. Some of the common ones that may be profitably adapted to modern data processing methods are outlined in the applications that follow.

General Application Areas

SALES

Sales are of vital importance to any business. Prompt, concise, useful reports are essential. Some of these reports are:

1. *Comparative Sales Volume* by product, by customer, by area, by salesman, or by any other desired

Types of Business

Utilities
Those companies concerned with providing widespread, essential products and services direct to a large-scale consumer market—electric, gas, and water utilities; telephone companies.

Institutions
Non-profit organizations of a public nature such as education, medical, and religious.

Financial Organizations
- Banks
- Brokerage Houses
- Investment Funds

Insurance
- Life
- Fire and Casualty
- Medical
- Auto
- General

Manufacturing

Construction

Transportation
- Rail
- Air
- Automotive and Motor Freight
- Ship

Distribution
Organizations concerned with high volume distribution of goods received in bulk.

Textile and Clothing Industries

Local, State, and Federal Governments

Agriculture

Entertainment
- Radio
- Television
- Movies
- Sports
- Resorts

Publications

Advertising

Retail

Figure A. Types of Business

breakdown. Figures have meaning when immediate comparison is given for sales this month to this month last year, for example.

2. *Commission Statements* detail sales force earnings and are part of the total sales expense report.

3. *Returns by Product* pinpoints defective products so that corrective action may be taken.

4. *Cost of Sales* reveals the profit margin of a product. Some of the required records in sales accounting are:

a. Cumulative sales for a period – year-to-date sales by product, customer, salesman, district, and so on.
b. Commission records.
c. Other sales expenses – advertising, meetings, travel, and so on.
d. Current transaction information – orders, adjustments, returns, and so on.

5. *Sales Forecasts* affect production scheduling, financial requirements, and so on.

Billing

A bill describes goods or services, prices, commodity numbers, customer information, and amounts; it is the source of many entries throughout the accounting structure of the business. The items on the bill appear in subsequent sales reports; the amount of the bill is added to accounts receivable; the items are deducted from inventory; and tax records are affected by tax calculations shown.

Pressure to produce bills on time is constant in any business because payment usually depends on receipt of a bill. Billing records include:

1. Customer identification and credit rating.
2. Tax information.
3. Item descriptions, codes, prices, discounts.
4. Type of service, rate structures, expiration dates.
5. Other data: method of shipment, terms of payment.

Accounts Receivable

Accounts receivable are the money due a company for goods and services rendered to its customers. The accounts receivable application records indebtedness and payments; a goal is fast collections so that the money may be more rapidly put to work.

The initiating records are the bills or invoices and sales adjustment data. Required records are:

1. Customer statistics and credit ratings.
2. Accounts accumulated to show past due or aged breakdowns.
3. Cash receipts records.
4. Credit memoranda.
5. Accounts receivable registers or ledgers.

Either of two basic types of accounting are practiced:

Balance Forward. Individual amounts are periodically summarized and added to previously summarized figures to give a balance forward lump sum.

Open Item. Each entry to the receivables file is maintained separately so that a statement to a customer will show each item rather than a single sum for all items prior to a certain date.

Inventory and Material Control

The accounting for quantity and dollar value of raw material and finished goods is an essential factor in cost control and good business operation.

Balance information on material or items is maintained by several categories: on hand, on order, requisitioned, returned, allocated. The effect of each transaction on the balances is reflected in the stock status report.

Through analysis of the reports, obsolete or slow moving items are eliminated from stock; material is ordered with enough lead time to prevent production slowdown caused by shortages; economical ordering quantities are considered when ordering stock replenishments to take advantage of price breaks; and the most efficient use of stock facilities is assured. Records required are:

1. Balances as of a certain date.
2. Code listings of each item by item number, type, class, and so on, depending on the extent of the analyses required.
3. Transactions affecting balances.
4. Stock status reports as balances change.

Accounts Payable

Accounts payable are money owed by a company to its vendors. The objective of accounts payable records is to keep account of debts incurred, to discharge those debts in time to take advantage of term discounts, and to keep account of expenditures in each area of the company.

A company's credit rating is established mainly by the speed with which it pays its bills. A properly functioning accounts payable procedure assures rapid payment and also highlights expenditures which may need review. Records required are:

1. Vendor information: name, address, purchases from this vendor by item and amount.
2. A file of items due for payment.
3. The payables distributions summary breakdown for ease of coding at the source.
4. Disbursement vouchers, expense vouchers, and so on.
5. Incoming invoice registers.

Payroll and Labor

These accounts cover compensation to employees for services rendered and distribution of labor costs. Records required are:

1. Year-to-date tax and earnings figures.
2. Master employee files.
3. Rate files, if incentive type payroll.
4. Time cards or sheets.
5. Payroll registers.
6. Statements and checks.
7. Payroll distribution reports.
8. Efficiency reports.

Manufacturing Control

This title is continually being broadened to cover new applications. Manufacturing control generally covers forecasting of machine and labor loads, scheduling products through a process or scheduling machines to do the processing, and complete planning for most efficient production.

Inventory control is tied in because total part and material requirements resulting from bill of material breakdowns must be submitted (if 8 flanges are needed for one frame and 10,000 frames are forecast, 80,000 flanges are needed). Payroll is tied in because labor requirements are forecast. Accounts payable, sales forecasts and performance, and plant and equipment accounting may also tie in to a manufacturing control application.

Some of the records required are:

1. Bill of materials: a breakdown of a product into its component parts.
2. Machine capacity figures.
3. Stock status figures.
4. Labor strength.
5. Market forecasts by product.
6. Progress reports of each production phase.
7. Shop orders.

The Total Systems Concept

When a system is analyzed for conversion to automated data processing, illogical boundaries between applications are sometimes assumed. An area, such as payroll, is analyzed and systematized without due consideration for the effects that transactions in this area have on other phases of the business. As other applications are converted, it becomes difficult to associate data from each in the most efficient and meaningful manner for a total presentation of the status of the business.

If a data processing system is to be most effective, long range goals and aspirations should be established in the beginning. A thorough shakedown of current procedures inevitably reveals weak points, duplication of effort, or excessive effort with too little return. Planning starts on firm ground when all the facts are in. When long range goals are known, application conversions may be undertaken with a future tie-in in mind. Equipment selection at this point may proceed with a more intelligent appreciation of present capacities and future needs.

Historical Records and the Audit Trail

Record keeping on a "don't throw anything away, we may need it later" basis is expensive and wastes space. Historical records, when properly kept, become part of an audit trail to allow re-creation, on paper, of the business at a point in time.

Because everything is relative, the present business picture is further illustrated by comparison with past records. "Sales this month" means little unless related to last month, or this month last year, and so on.

An audit trail is used by management more often than by the auditor. Any time management requests proof of a report, the audit trail is referenced. A properly designed data processing system will also satisfy the auditor's needs.

In its simplest form the audit trail consists of summary information as of a certain post accounting date plus all transactions affecting that information up until the date of the audit. The auditor should be able to request detail information supporting any accounting entry in question. A formal audit of a company's books is usually arranged on a cyclical basis. Because it is physically impossible to examine everything, spot checks are made in each area being audited. Because the firm being audited does not know the specific area the auditor will question, it can be as-

sumed that the books are in order if those areas covered are satisfactory.

The nature of a modern data processing system is such that information required by an auditor is presented logically and clearly. The auditor's requirements parallel those of a well designed system.

A Modern Management Approach

Management By Exception

Managers are busy people. They have little time for routine. The knowledge and judgment justifying the title of manager are ill used for reviewing normal daily activity. Characteristic of most management personnel is an ability to strike through detail to the core of a problem. A data processing system can assist management by selecting only items requiring review. Much of the detail is eliminated so that judgment is applied where it is needed with no wasted time.

For management by exception to be effectively employed, the boundaries marking normal and abnormal data must be set up when the data processing system is planned. For example: In a billing and receivables application, each customer is given a credit limit by the credit manager. Purchases are charged up to that limit. Whenever the limit is exceeded, a notice is produced by the system for the credit manager's attention. Management by exception thus conserves management prerogatives. Automatic refusal to approve purchases beyond the limit could be programmed in the system, but the consequences of such an inflexible policy could be ruinous.

Management by Projection

Shortly after large scale data processing equipment appeared on the market, (and in a few cases before), uses beyond the conventional became apparent. Perhaps the best known of these are simulation techniques.

To simulate something on a computer requires that the thing to be feigned be well defined. Its desired and known characteristics are set down in as much detail as possible. The variables are then introduced and altered as often as required to produce a result closely approximating or exactly matching a predetermined objective. This is trial and error at high speed. A good example is simulation of an airplane in flight and the effect of design on performance characteristics. The ability to try thousands of design variations before the plane is even off the ground has obvious advantages.

Engineering design problems are well suited for simulation methods because definitions of factors and variables is fairly precise. When general business problems are applied, however, the necessity for clear definitions and proper weighting of factors requires the application of management science principles. Here is where the businessman-mathematician fits.

A company considering the opening of a new branch in another geographic area may, by simulation, test many different locations to pick the most advantageous one. Or, internally, a company considering a new product or a procedure change may test the effect of the change at each production stage before committing itself. The potential for application of management science techniques is growing.

Changes taking place in the data processing field have far-reaching implications. For one, the trend toward decentralization may be reversed. With management science and equipment to match its requirements, a centralization of operations and its logical economies is again practical. Much of the decentralization of large organizations in recent years occurred as a result of a negative philosophy imposed by communications breakdowns. It was difficult for management to manage because information was hard to obtain due to sheer organizational size. If operations research and management science teams can enlarge upon the ingenuity they have already demonstrated, a wealth of useful data will be available. Managerial responsibilities may then tend to return to the top level, well informed executive as middle (branch or divisional) management relinquishes some of its powers. It is reasonable to assume a broad impact in business structures as new data processing and information handling techniques develop.

Index

	Page
Access Arm	33
Access Time	29, 34, 46
Accumulator Register	37
Adder	38
Address, Instruction	56
Address Modification	69
Address Register	37
Arithmetic-Logical Section, CPU	15, 37
Array, Core Storage	31
Assembly Program	74
Autocoder	75
Automatic Coding System	75
Auxiliary Operation	54
Auxiliary Storage	29
Batch Processing	9, 34
Binary Card	24
Binary Coded Decimal	20
Binary Mode	19
Binary Notation	19
Binary Number System	20, 23
Binary Tape, Magnetic	27
Bi-quinary Code	22
Bit	19, 20
Bit Density	46
Block Diagram	57, 58
Branch Operation	39, 66
Buffer, Data	54
Calculating	64
Card Code	24
Card (IBM)	17, 23
Card Punch	44
Card Reader	43
Carry	38
Cathode Ray Tube	52
Central Processing Unit	15, 36
Chain Printer	51
Character Code Check	21, 22, 48
Character Rate, Magnetic Tape	46
Check Bit	21, 22
Check Character	27, 47
Check Point and Restart	88
Code	20
Code Check	20
Column Binary, Card	24
Compare Operation	68
Compiler	80
Computer Codes	20
Console	16, 52
Control Panel	41
Control Section, CPU	15, 36
Control Total	87
Core Plane	31
Core Storage	30
Counter	38
Crossfoot Checks	88
Data Buffering	52
Data Channel	54
Data Conversion	18, 54
Data Processing	5, 12, 15
Data Representation	17, 18
Density	46
Disk Storage	33
Display Lights	16
Drum Storage	32
Eight Channel Code, Paper Tape	25
End-of-File	46
End-of-File Gap	46
End-of-Reel Marker	49
Entry Hub	43
Execution Cycle	39
Execution Time	39
Exit Hub	43
File	34, 64, 69
File Protection Device	49
File Reel	44
Five-Channel Code, Paper Tape	26
Fixed Count Check	21
Fixed Word Length	40
Flag Bit	22
Flow Chart	57
Fortran	82
Horizontal Check, Magnetic Tape	27, 48
Housekeeping Checks	88
Indexing	70
Indicator	19
Indirect Address	71
Inhibit Wire	32
In-line Processing	9, 34
Input	12, 13
Input Devices	13, 18, 41
Inscribing	28
Instructions	12, 38, 56, 57
Instruction Counter	38
Instruction Cycle	38
Instruction Modification	68
Instruction Time	39
Inter-record Gap	46
Library Program	74, 79
Limit Check	88
Literal Operand	76
Load Point Marker	48
Logical Operations	65
Longitudinal Check, Magnetic Tape	48
Machine Checking	89
Machine Coding	73
Machine Cycle	38
Machine Language	73
Machine Operations	15
Machine Reel	44
Macro-instructions	78
Magnetic Core	30
Magnetic Disk	33
Magnetic Drum	32
Magnetic Ink	27
Magnetic Label	88
Magnetic Tape	26
Magnetic Tape Unit	44
Main Storage	29
Memory (See Storage)	
Mnemonic Code	75
Numeric Bit	21, 22
Object Program	74, 79
Off-Line	14, 54
On-Line	14
One-Gap Head	45
Operand	38, 56, 76
Operating Keys and Lights	16, 41
Operation (See Instruction)	

Page

Output 12, 13
Output Devices 41
Overflow 37
Overlap Operation 52

Paper Tape 25
Paper Tape Punch 50
Paper Tape Reader 49
Parallel Operation 40
Parity 20
Parity Check 21
Permanent Storage 35
Photoelectric Sensing 43, 48
Photo-sensing Markers 48
Print Wheel 50
Printers 50
Procedure Control 85
Processor 74, 78
Program 15, 56
Program Assembly 75
Program Compilers 80
Program Development 57
Program Language 74
Program Loop 65
Program Package 81
Program Preparation 72
Program Routine 81
Program Switch 69
Program Systems 73
Proof Figures 87

Random Access 34
Reading Data 64
Read-Write Head 44
Record 35, 46, 66
Record Block 47
Record Count 87
Reflective Spots 48
Register 37
Row Binary, Card 24

Secondary Storage 29

Page

Sense Wire 32
Serial Operation 40
Seven-Bit Alphameric Code 21, 27
Shifting Operation 37
Signal 18, 19
Six-Bit Numeric Code 22
Solid State Components 9
Sort Programs 83
Source Program 74
Storage 14, 29, 34
Storage Register 37
Stored Program 15, 56
Sub-routine 66, 70
Symbolic Language 78, 82
System Checks 86

Tag 74, 75, 78
Tape Label 88
Tape Mark 46
Tape Records 46
Track 27, 33
Transfer Operation 39, 66
Transient Storage 35
Two-gap Head 45, 47
Two-Out-of-Five Fixed Count Code 21
Typewriter 16, 51

Utility Programs 84

Vacuum Columns 44
Validity Checks 41, 47
Variable Word Length 40
Vertical Check, Tape 47

Wire Matrix Printer 50
Word 23

Zone 21, 24, 25
Zone Bit 21

F22-6517

Printed in U.S.A. F22-6517

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York