IBM

General Information Manual

705 Data Processing System

General Information Manual

IBM 705 Data Processing System

Preface

The use of electronic computing machinery in handling the gigantic paperwork problems of today is becoming an almost commonplace occurrence. Newspapers, magazines, trade journals, and popular advertising have made the term "giant brain" a household word. In less than one decade, the work accomplished by computers has begun to affect every aspect of modern life. Computers design highways and track space missiles; they assist in the engineering of outboard motors and atomic power plants; they do the bookkeeping in the smaller commercial enterprises and the accounting work for the federal government. As the value and practicability of computers is being proved, new fields are being entered almost daily.

This manual is intended as an introduction to the IBM 705 I, II, and III Data Processing Systems. The manual is divided into two main sections. The first deals with the system as a whole and describes in somewhat general terms the operation of core storage, various registers and accumulators, drum storage and other input-output devices. Complete detail has been intentionally omitted. Much of this presentation can be applied to computers in general, as well as to the 705 in particular.

The second part of the manual is an introduction to 705 programming. A comparatively simple problem is discussed from its definition to final solution. All of the various data handling and calculating functions which the machine can perform are summarized with schematic diagrams of the data flow.

A brief introduction to the autocoder is also given. The autocoder is an approved and simplified method of programming for the 705. Throughout the manual, intended emphasis is on what the machine does, rather than how it does it. Further details may be found in the *IBM 705 Data Processing System Reference Manual,* Form A22-6506.

Contents

IBM 705 Data Processing System

IBM 705 Data Processing System

The diagram (below) shows one type of application for a 705 in a manufacturing plant.

Two kinds of basic facts, manufacturing and sales, flow into the data processing system. Flowing out of it is information (obtained rapidly and systematically) that:

1. Cuts down record keeping and report writing.
2. Supplies up-to-the-minute operating data.
3. Relieves management of many routine decisions.
4. Compares results with predesigned standards.
5. Enables operating executives to make decisions based on facts.

The practical day-to-day use of IBM 700 systems is an accomplished fact. The computers, first designed for complex scientific and mathematical analyses, have been modified to solve plant operating and management problems. They are capable of providing more information and better control than ever before.

It is significant that the use of such systems is not restricted to the corporate giants. Smaller plants are also able to use scaled-down models of the system, tailored to their needs. They may also buy time from a service bureau at an hourly rate. The IBM 705 can be expanded to keep pace with the growth of a business.

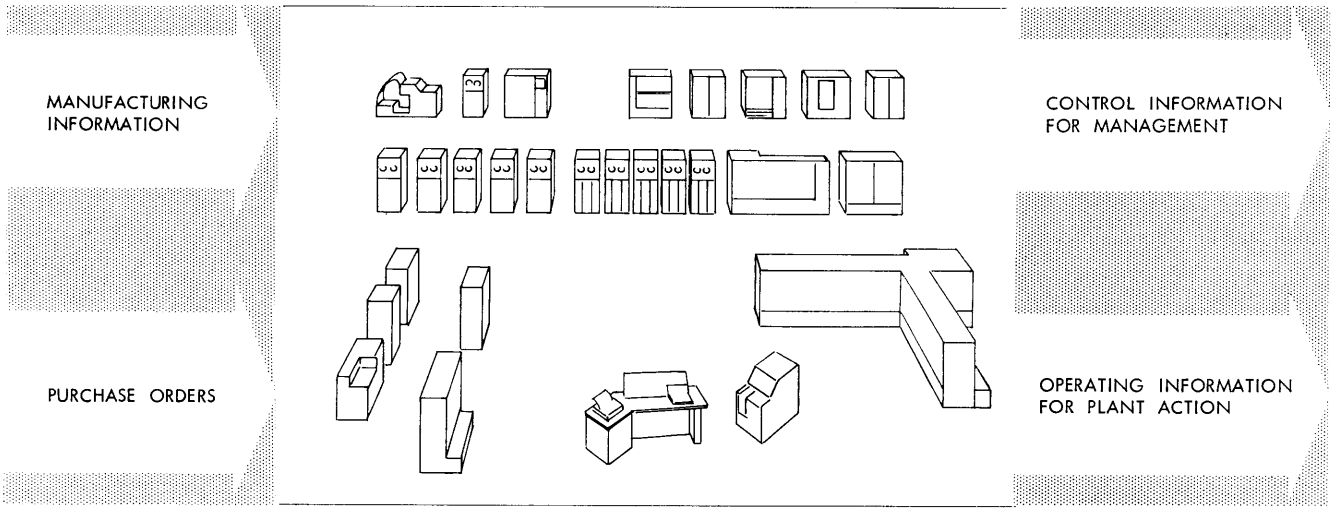
The 705 digests data in huge quantities at high speed according to required procedures. Results can be either detailed or summarized. The machine stores instructions, analytic routines and decision-making procedures as well as data. It can flag exceptions, errors in original transactions and illogical answers. It can combine in one pass of data through the system what has been formerly accomplished with whole batteries of punched-card or other office equipment. It reduces document handling, combines records, and puts files on reels of magnetic tape. Its highest pay-off may be on jobs that other equipment cannot do. New applications are being developed almost daily.

The 705 does seven basic data-handling jobs. It can do any one of these alone, or several at the same time. The more jobs that can be finished in one processing of the records, the more efficient the application of the machine.

For example, in a materials control procedure, assume that the 705 is applied to the control of a critical component (in this case, switches). In the same way, it can also be applied to many other plant operating problems.

File Maintenance

File maintenance is an inventory job with records. The file may represent parts in stock, back orders, maintenance costs, job costs, quality records, or work in process. Using tape or drum, the IBM 705 not only acts as an electronic file clerk, but adds, subtracts, and computes balances automatically. Result: current records that reflect the true status of the business.



IBM 705 — Manufacturing Application

For instance, millions of type x switches may be used. The 705 improves both inventory control and production planning. In addition to complete data about type x switches, an up-to-date stock status is always kept. At regular intervals, receipt or withdrawal transactions are fed into the machine. In this way, the latest facts about switches are available, either directly from the 705 or from reports printed by auxiliary equipment.

Process Volume Paper Work

The IBM 705 makes all calculations for invoices, parts lists, production schedules, shipping manifests, job costs; the higher the volume of documents processed, the higher the savings in cost. Result: no clerical drudgery, no transcription errors.

One comparatively simple job for the 705 is to print a list showing all switches needed for scheduled production. A more complex job would be to compare the inventory file with parts-requirements files to show how many switches to order. A still more complex job would be to compare inventory with requirements, automatically print purchase orders, and address them to vendors. The machine might also acknowledge receipt of the order and write a check in payment, if quality and quantity are satisfactory.

Communicate with Other Machines

The IBM 705 communicates directly with existing punched-card equipment. It can receive data either directly from cards or from magnetic tape produced from cards. Cards can also be converted to punched paper tape, and vice versa. Either cards or tape can be transmitted over wire.

For example, with leased wires, inventory control can be centralized for remote plants or warehouses by receiving scattered reports, consolidating them, and transmitting combined results back to their source. The 705 can also use high-speed printers, an electric typewriter, card punches, and standard card-operated machines. Result: fast communication into and out of the computer system; effective inventory and production control with minimum clerical paper work.

Several direct input devices can be used to feed data to the 705 about switches. An output printer can be used to turn out all finished paper work such as purchase orders and control reports.

Make Decisions

The IBM 705 can compare production costs with predetermined standards and print out items for which costs are out of line with standards. With this information, corrective action can be taken quickly. Stock items can be reordered automatically when they drop below preset limits. Back orders, overhead schedules, and maintenance can be balanced with available hours of shop operation. Result: no scurrying to find facts on which to base decisions about ordering and scheduling.

Modify Basis for Decisions

The IBM 705 can be instructed to change specifications on a comparison basis as significant facts change. Thus, it can adjust costs to current information and keep comparison standards up to date. It can continually recalculate values to shift reorder points, time or cost standards, lead times. It can adjust for the changing cost of borrowed money. It does all this automatically, computing with latest values. Result: effective management control.

Combining jobs, the 705 alters the low limits on switch inventory as consumption changes. It scans all costs that affect ordering policy and decides on order quantities and buying practices that keep inventory at the most efficient level.

Compare Alternatives

The IBM 705 solves linear programming and operations research problems, weighing a number of possibilities to show the best. It can examine a number of product mixes for profit potential and pick the one for maximum profits or look at a number of machine tool combinations and pick the best. Result: quick, sure decisions among many courses of action.

The 705 has been told to weigh prices, raw material availability, shipping distance from various vendors, labor costs, and so on. When switches are needed, the machine examines all factors, determines which vendor to order from, or whether to make switches in the company's shops or split orders among several vendors.

Solve Formulas

The IBM 705 can be used to solve mathematical problems. It does calculations such as economic lot size for production or profitability in a materials-control plan. It can also speed solution of the many management problems that require a high degree of scientific analysis. Result: sure answers to problems that would keep a battalion of clerks busy for months, possibly without their being able to solve them at all.

The 705 can develop and use new sophisticated formulas that human minds cannot handle. These

formulas can be used to calculate economical ordering quantities and even to predict future needs from present consumption trends. And, by including all costs in such a formula, orders can be spotted that will lose money, in time to turn the orders down. An entire business operation can be simulated mathematically, showing the results of future planning in advance without costly trial and error.

The 705 can also be used for engineering and research. For example, it can solve flight trajectory problems, analyze stresses on materials, or calculate lift and drag for wing surfaces.

IBM 705 Language

The IBM 705 is a machine system that can process information. It is primarily designed to handle the type of information originating either in commercial or in government procedures (military, statistical, or other related areas). It may also readily be adapted to scientific calculation, particularly where the volume of data approaches that found in commercial work.

All such usable information originates first as written records of facts pertaining to the operation or state of an enterprise. The records assume various forms: paper documents, punched cards, punched paper tape, ledgers, and all types of files. And normally, no matter in what form the records are kept, the data are represented or coded in the symbolism of numbers, letters, and characters. In business, these familiar symbols stand for an endless variety of quantities, values, descriptions, categories and types of goods, people, services, volume, property, and so on.

There is no language without some numerals.

It is almost universally accepted that the wide use of the decimal system originated from the habit of counting on ten fingers. In some languages, the same word is used for "ten" as for "two hands." The word "digit" refers either to one of the numerals 0 through 9, or to a finger or toe.

If the decimal system came into being to facilitate counting on fingers, it is not at all surprising that this system might not be the best one for electronic computers. The computer has only two "fingers," that is, nearly all present components are inherently binary (Figure 1). An electro-magnetic relay maintains its contacts either closed or open. A magnetic material is utilized by magnetizing it in one direction or in the opposite direction. A vacuum tube is maintained either fully conducting or non-conducting.

The plus or minus voltage conditions of specific circuitry can be used to indicate yes or no, on or off, one or zero, plus or minus, and so on. While reading or writing, the machine can also sense punched holes in IBM cards. The presence of one or more holes indicates information. The absence of punching indicates no information.

A computer, therefore, operates most efficiently with a language of its own based upon the binary nature of its components.

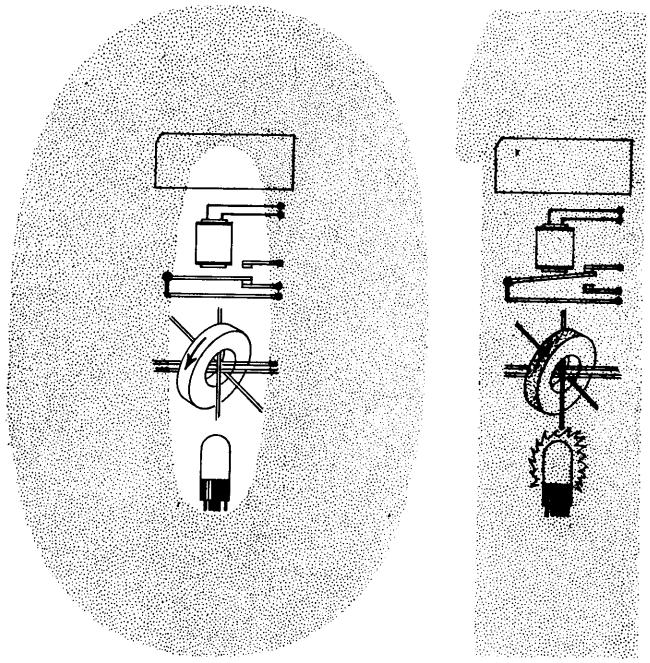


Figure 1. Binary Nature of Components

Since it is hardly practical to keep business records in some binary form of notation, data must first be converted to some standardized form (punched cards, for example) that the machine can read and then translate into its own binary language. The machine should then be able to write in this standard form (punch out cards) for reconversion to written records, or it should be able to print these records directly in usable form with no conversion. The 705 does both.

To keep the translation process simple, the binary representation within the machine should be compatible with all common symbols used in business. The machine will then have the ability to manipulate both alphabetic and numerical information as well as special characters. Furthermore, it will have the advantage of being able to calculate using binary arithmetic — providing an additional factor of speed and efficiency.

To meet these requirements, the 705 converts all information to a modified binary representation of numbers, characters, and symbols as it first enters the system. As explained in the following sections, the conversion to this language is made automatically from IBM punched cards.

The IBM Card

Before the recent development of magnetic tape, the punched card was almost universally accepted as the most practical device for recording information for machine processing. It is still the basic unit record form used by all IBM card-operated equipment.

All data in punched cards are represented by punching small rectangular holes in predetermined positions (Figure 2). The card is divided into two main areas: the lower (numerical) section and the upper (zone) section.

The numerical section is further divided into ten horizontal rows, one row for each digit 0-9. The zone section is divided into three horizontal rows: 0, 11, and 12. Note that the zero row is common to both zone and numerical sections.

The standard IBM card is also divided into 80 vertical columns. To record data, a character is represented by punching one or more holes in a single column. Thus, as many as 80 characters may be punched in one card.

Holes punched in their proper rows and in specified columns can be automatically identified as characters by IBM card equipment or by the card reader attached to the 705. Digits are represented by single holes punched in the numerical sections; letters and special characters, by combinations of zone and numerical punching.

For example, the 12 zone with one of the digits 1-9 represents the letters A-I; the 11 zone with the digits 1-9, the letters J-R; the 0 zone with the digits 2-9, S-Z. Eleven special characters are represented by either single zone punches or combinations of zoning and multiple digit punching. This system is often referred to as the "Hollerith Code" after Dr. Herman Hollerith, who first developed machines operating on this principle.

Binary Coded Decimal

The binary, or base two, number system uses just two symbols (0 and 1) to represent all quantities. The place value of the symbols is based on a progression of multiples of 2. For example, the units position of a binary number has the place value of 1; the next position, a value of 2; the next, 4; the next, 8; and so on. Under this system, the quantity of 12 is expressed with the symbols 1100, meaning $(1 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$ or $(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0)$. Again, the various orders do not have the meaning of units, tens, hundreds, thousands . . . as in the decimal system. Instead, they signify units, twos, fours, eights, sixteens, and so on.

The following table shows the binary representation of numbers from 0 to 9 used by the IBM 705.

DECIMAL	BINARY 8 4 2 1 (Place Value)
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
0	1 0 1 0

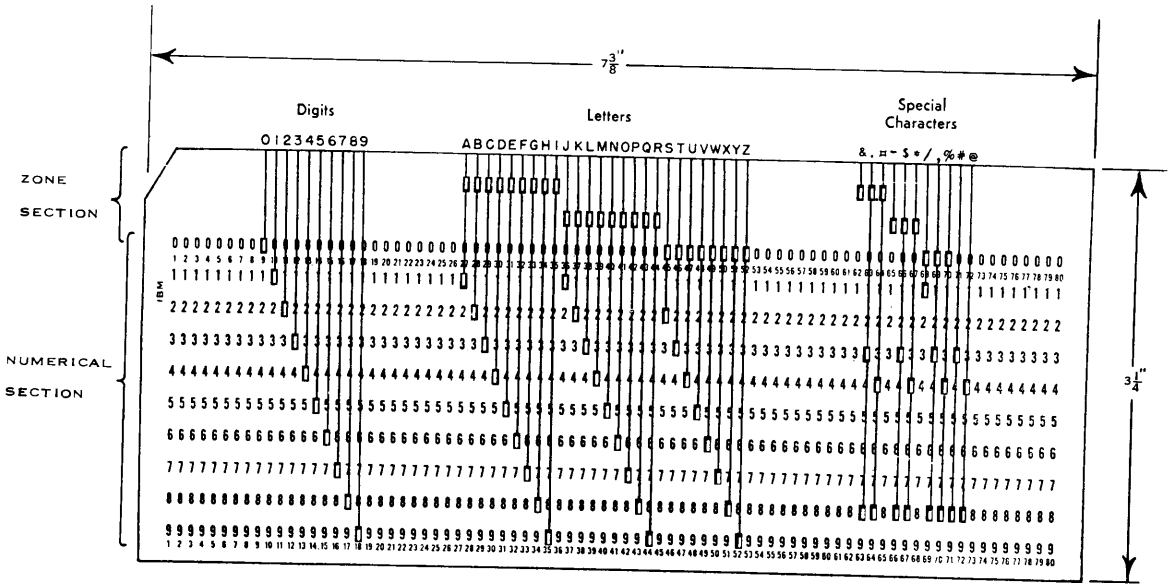


Figure 2. IBM Card Character Coding

Note that 0 is represented as 1010, actually the binary number for ten.

The 705 representation of characters divides the binary code into two sections in the same manner as coding is divided in the IBM card:

1. The three zones, numbered in binary 01, 10, and 11.
2. The numbers 1-9 as binary 0001-1001. The 0 is represented as 1010.

The code, called binary coded decimal, consists of six binary positions divided as follows:

```
00 0001  through  00 1010 = 1-9,0
11 0001  through  11 1001 = A-I
10 0001  through  10 1001 = J-R
01 0010  through  01 1001 = S-Z
```

Other combinations of the zones and numerical digits represent special characters. Still other combinations form special marks and symbols used by the 705 to mark the end of records, fields, or blocks of data.

The BCD code also uses a seventh position to provide a check on the validity of all information handled by the machine system. A single check position is added to the code so that the total number of 1's used to represent any character, including the check, is always even.

For example, if the code for the letter L is 10 0011 (using an odd number of 1's), a check 1 is added to make the total even. Consequently, the complete BCD code for L is 1 10 0011. If the code for the letter M is 10 0100, no check 1 is added. The complete binary code for M then becomes 0 10 0100. The zone and check positions of the code are also referred to as the A, B and C positions.

Figure 3 shows the BCD code for digits, letters, and special characters as used by the 705.

		0 1 2 3 4 5 6 7 8 9	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	& . ¤ - $ * / , % # @
Check	C	0 1 1 0 1 0 0 1 1 0	1 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 0 1	0 1 0 1 0 1 0 0 1 1 0
Zone	B	0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0	1 1 1 1 1 1 0 0 0 0 0
	A	0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 0 0 0 1 1 1 0 0
Numerical	8	1 0 0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1	0 1 1 0 1 1 0 1 1 1 1
	4	0 0 0 0 1 1 1 1 0 0	0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0	0 0 1 0 0 1 0 0 1 0 1
	2	1 0 1 1 0 0 1 1 0 0	0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0	0 1 0 0 1 0 0 1 0 1 0
	1	0 1 0 1 0 1 0 1 0 1	1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	0 1 0 0 1 0 1 1 0 1 0

Figure 3. Binary Coded Decimal

Storage

All conventional calculating machines and related office equipment have ability to store information. Even the smallest adding machine or desk calculator stores numbers and their sums. Usually, the numbers are entered by some key-driven mechanism.

The idea of storage for numbers, quantities, and results is not new. The development of calculating machinery has depended, to a large extent, upon the improvement of various types of storage devices. In earlier times, many attempts were made to provide mechanical devices to accumulate quantities and store results. Perhaps the most successful and widely used of these is the Chinese abacus.

The accumulation of any information by a calculator (including the abacus) is the result of a number of predetermined operations carried out in a prescribed sequence. For example, to use an adding machine, the operator first enters the numbers by pressing keys; then he adds or subtracts this quantity in a counter (storage) by pressing a special key. To accumulate results, the routine is repeated as many times as there are quantities to add. To get a total, another key is pressed. Depending upon the type of machine, other keys take subtotals, list without adding, repeat, and so on.

The concept of automatic control over the sequence of calculator operation is also not new. A complex computing machine, found in a dusty storage room of a Greek museum, is believed to have been used to compute planetary orbits as early as 65 B.C. It contained complicated groupings of gears, a series of calibrations, and Greek inscriptions explaining the theory of the machine and various cycles of the sun and moon. In 1833, Charles Babbage, a British mathematician, conceived of an "analytical engine" to have flexible sequential control over the operations it performed. Babbage designed his machine (the real ancestor of modern computers) so that it would be possible to specify in advance the sequence of calculation and the numbers to be operated upon. Control was to be effected by a punched card mechanism developed earlier for the Jacquard loom. Also, the sequence could be altered by changing the punched cards. Numbers were to be stored by mechanical wheels. Unfortunately, this brilliant conception was never translated into a working machine, partly because of financial difficulties and partly because of engineering problems that were insurmountable at that time.

Until 1945, all calculating machines that might be classified as "automatic" used some external means of sequence control: punched cards, paper tape, wired control panels, or combinations of each. Storage was used only to store numbers. The revolutionary concept of also placing instructions in storage first appeared in a report written by Dr. John von Neumann, who proposed a computer using this principle. By storing the instructions internally, and by using binary instead of decimal numbers, von Neumann foresaw that much greater power would be achieved at considerably less expense of electronic equipment.

Thus, one of the most important differences between the modern digital computer and the more familiar calculators and accounting machines is the computer's ability to store its instructions as well as data. The computer can be said to "remember" what it is told to do and to "recall" its instructions as needed. Since the functions of storage may be compared, in a limited sense, to the functions of the human brain, computer storage is often referred to as "memory."

The larger the storage or memory capacity and the faster it can store and recall, the more versatile is the machine. When information is read, it too is temporarily stored in memory. It can then be acted upon according to given instructions in the proper sequence. Again, results are stored until ready to be sent to some output unit: tape, printer, or punch. In addition, if memory is large and fast enough, entire tables of rates, codes, and other reference material can be held ready for instant use. Cross references between incoming records and tables are made inside the computer, at electronic speed. Time for searching through code books or rate tables is cut to an absolute minimum.

If there is too much reference material to remember, the machine can search a reel of tape for the proper section of the table, then bring this into storage. Next, the section is examined more closely for the proper item — somewhat like looking through a book, one chapter at a time.

Another way to have a larger memory is to attach a secondary storage device, a magnetic drum, to the main memory. Tables (or other material) are transferred from tape or cards to the drum before operation begins. When needed, the drum is searched at much higher speed than tape (but also at higher cost of equipment). A drum or tape is sometimes used to

store instructions, preferably only those used when exception to normal processing is encountered.

The 705 has a large memory. Available models can store 20,000, 40,000, or 80,000 characters. Characters are defined as letters of the alphabet, digits, and a number of special symbols commonly used in business, including @ # & % ☐ / * − $, Provision is also made to attach magnetic drums to supplement storage capacity. Each drum can store as many as 60,000 characters.

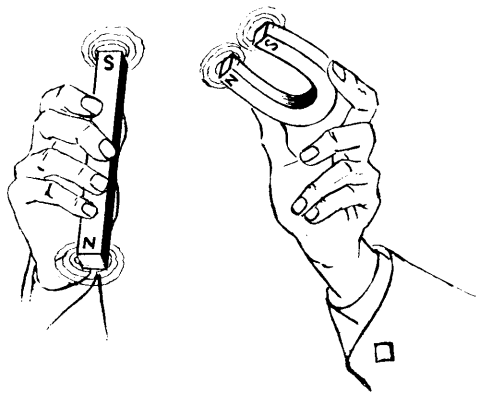


Figure 5. Polarity of Magnets

How Magnetic Cores Remember

All information handled internally by the IBM 705 — data, records, or instructions — is handled in binary coded form. As previously explained, this is the character representation used in the central processing unit.

While such a coding system would be clumsy for paper-and-pencil work, it is considerably easier for the machine to use. This is because the more than 50 different characters used to write data can be represented as binary numbers and (the real advantage for the machine) any character can be coded merely by setting up the proper combination of 1's and 0's. Actually, only two numbers, 1 and 0, are represented in memory, rather than 50 different symbols, letters, or numbers.

A simple counter wheel represents a quantity or value by degree of rotation and its position with regard to other wheels in the counter (Figure 4). In the 705, the two values 0 and 1 are represented by a device that can be conditioned to assume one or the other of two states. The device is the magnetic core. The two states can be thought of as yes or no, on or off, positive or negative. The positive state can be made to represent 1; the negative, 0.

A magnetic core is a tiny ring of ferromagnetic material. When seen, it appears to be a miniature doughnut about half the diameter of a match head. Like a doughnut, each core is pressed from a powdered batter (ferric oxide and other materials) and then baked in a high-temperature 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, an additional factor of reliability.

The common bar or horseshoe magnets show a definite polarity. That is, one end of the magnet is thought of as having a north pole; the other, a south pole (Figure 5). However, the polarity of a core may be readily set up in either direction, depending upon the direction of current which induces the magnetic state. Thus, a core can be magnetized in either of two ways which can be described as positive or negative. These are the two states needed for storage: a positive core can represent a one; a negative core, a zero.

If cores are placed on a wire — like beads on a string — and a strong enough current is sent through the wire, the cores become magnetized (Figure 6). The

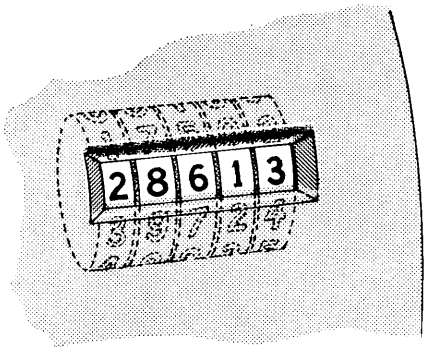


Figure 4. Counter Wheel

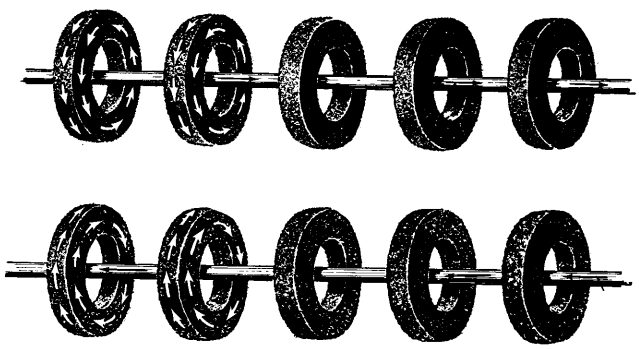


Figure 6. Polarity of Magnetic Cores

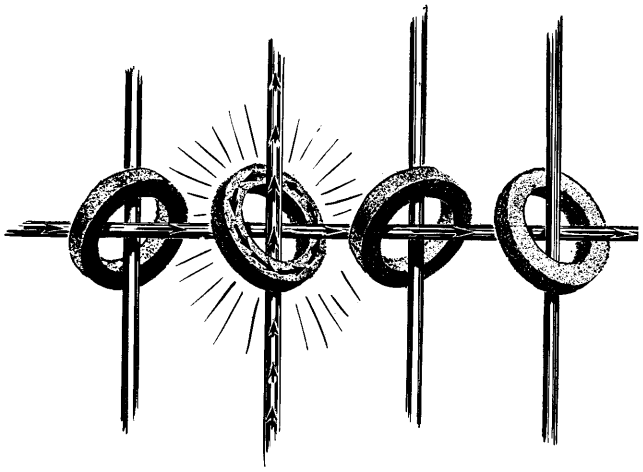


Figure 7. Core at Intersection of Two Wires

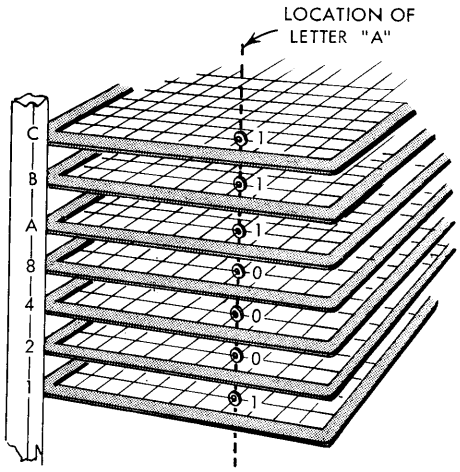


Figure 9. IBM 705 Memory Location

direction of current determines the resulting polarity of the cores. By reversing the current, the polarity is reversed.

In its largest memory, the 705 uses well over half a million of these small elements. Since any specified location of storage must be instantly accessible, the cores must be arranged so that any combination of ones and zeros representing a character can be written magnetically or "read" back when needed.

To accomplish selection, another wire is run through each core at right angles to the first (Figure 7). When *half* the current needed to magnetize a core is sent through two wires, 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 referred to as a "plane" (Figure 8).

To illustrate the use of a number of planes to store a character, assume that the letter "A" is to be placed in memory. To conform to the binary coding system of the 705, seven planes are needed: one for the check position of the character, two for the zone portion, and four for the numerical portion. One core in each of the planes is magnetized positively or negatively to represent the binary configuration for the letter A, 1 11 0001 (Figure 9).

Notice that with the planes stacked in an "array," the cores representing A are all 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 actual physical location of one character position of memory.

In the 705 I, 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 memory positions. (It passes through 35 cores, seven cores per character, consequently five positions) (Figure 10). Thus, the capacity of the 705 I memory

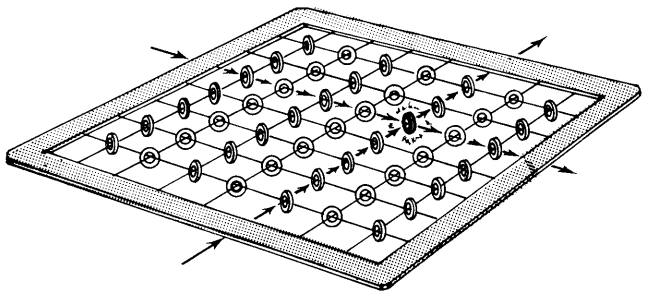


Figure 8. Magnetic Core Planes

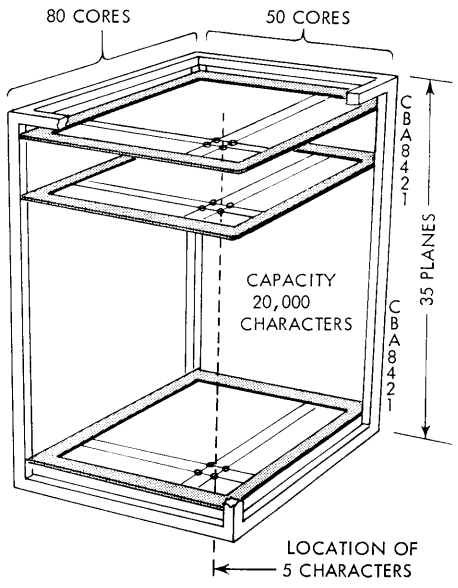


Figure 10. Schematic, 20,000-Position Memory

is 20,000 characters. Internal storage in the machine is always handled in five-character or 35-bit words regardless of whether one or more characters are involved. (This is of no concern to the user, because for his purpose any single memory position is available.) In this respect, the 705 is a parallel, fixed-word-length computer and closely resembles the 704 and 709. The similarity does not hold, however, for arithmetic and other operations.

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 may be said to "flip" the core to positive or negative, depending upon 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 third wire also running through the center of the core (Figure 11). 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, since 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 12).

Notice, however, that when information has been read from memory, 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 memory, the machine must replace 1's in those cores which had previously contained 1's. But cores which contained 0's must remain as 0's.

To reproduce ("regenerate") the 0's and 1's as they should be, the 705 tries to write back 1's in *all* the locations previously read (35 cores) but at the same time, an "inhibit" pulse suppresses writing in cores that previously contained 0's. The inhibit pulse is

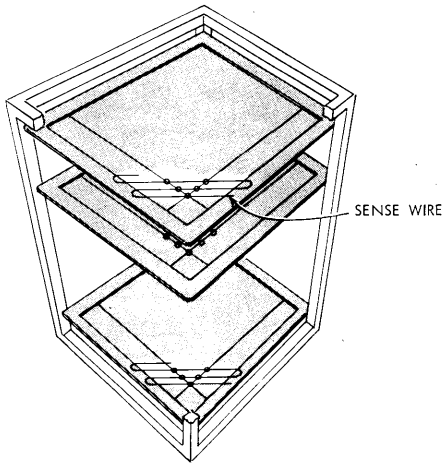


Figure 12. Sense Wire in Core Plane

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 also runs through every core in a plane (Figure 13).

It is beyond the scope of this book to explain fully the storage circuitry. However, a basic knowledge of how core storage works is useful in learning the operating principles of the 705. It is also useful in understanding the operation of other related machines using a core memory.

Some additional facts about 705 memory may now be summarized:

1. Access time to memory is nine millionths of a second. During this time, a 35-bit, five-character word is either stored or read out and regenerated. The same basic cycle is taken for both storing and read-out.

2. The time required to get an instruction from memory is 17 microseconds in the 705 I and II, 13 microseconds in the 705 III. This includes the time to read the instruction, replace it in memory, and decode it.

3. All instructions are five-character words that fit the 35-bit configuration of storage. Reading of instructions, therefore, takes full advantage of the parallel

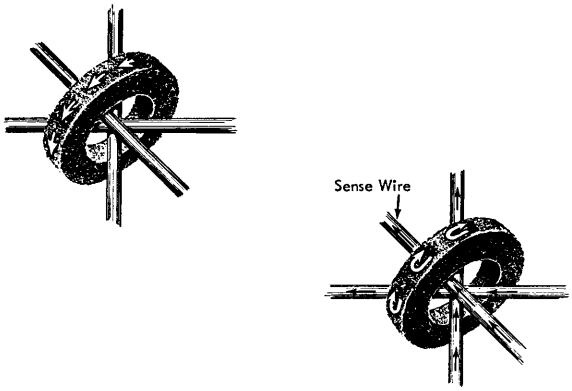


Figure 11. Core Sense Wires

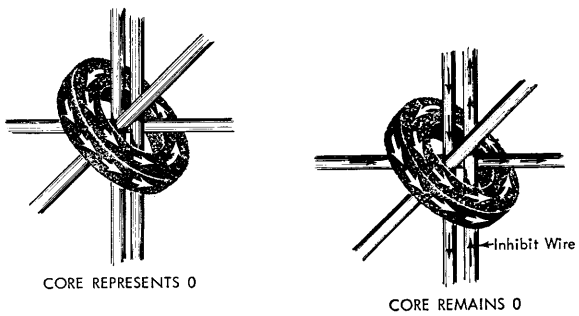


Figure 13. Core Inhibit Wire

method of moving binary coded information in and out of memory at extremely high speeds.

4. During machine operation, information stored in memory remains there indefinitely until other information replaces it. Old information is automatically erased whenever new information is placed in the same storage locations.

5. Since core storage is permanent, information once placed in memory should remain there even after the power is shut off. However, when power is turned on, various voltages are brought up sequentially to the proper level. The resulting surge of power throughout the system tends to change the state of the cores. Consequently, one of the functions of the power-on key is to reset all of memory to blank characters to make sure no unwanted information is present when the operation begins.

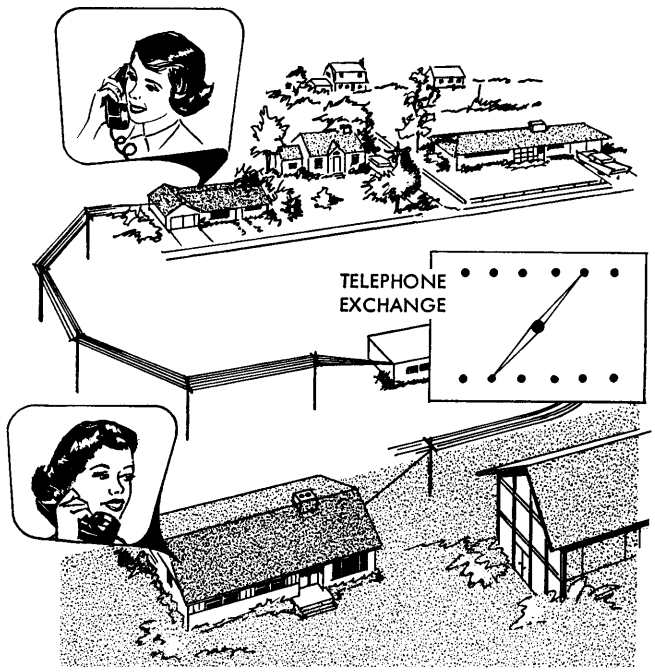


Figure 14. Telephone Exchange System

Arithmetic and Logical Unit

Nearly all of the operations within the computer are performed by transferring data from their memory locations, through a control or processing unit, and back to memory. Addition, for example, is performed by transferring one of the two quantities to be added into an accumulator. The sum is formed in an adder and transferred back to the accumulator where it is available for storage in memory.

It is the function of the arithmetic and logical unit to determine and control the route of these data transfers. Since the machine can operate only under control of instructions, it is also the function of the arithmetic and logical unit to select the instructions from memory, interpret these instructions and to control the over-all operation of the computer system.

In many ways, the arithmetic and logical unit can be compared to a telephone exchange system (Figure 14). All of the data-transfer paths that might be used in any operation must already exist, just as there must be connecting lines available between any two or more telephones. There must be a path or wire between every two units in the 705 that will ever be connected; there must also be devices that are capable of opening and closing these paths. These devices are often referred to as gates. As the name indicates, a gate opens to allow electrical pulses to pass through and closes to prevent them from using the path.

The function of interpreting an instruction involves opening and closing the proper gates for the given operation, just as connecting the proper lines allows conversation to take place between two distant telephones. The control circuits, then, must set up a different combination of gates for every instruction (Figure 15).

Each procedure involves a series of instructions called a program. The program is a completely detailed set of orders to the machine, telling exactly what

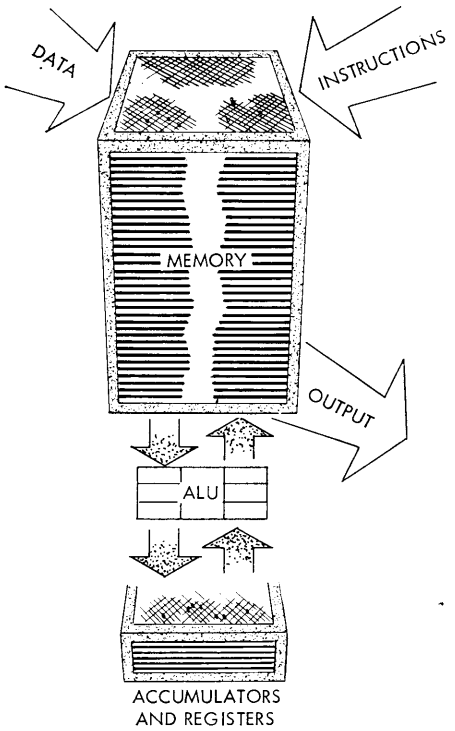


Figure 15. Schematic, ALU, Memory, and Accumulator

is to be done and in what sequence. The arithmetic and logical unit operates as directed by the program, one instruction after another. As each new instruction is brought in, the necessary circuitry is actuated to perform that instruction.

The Programmer

The programmer is the person who actually puts the computer to work. He produces the necessary link between the machine and the problem or procedure it is to do. Therefore, he must be thoroughly familiar with the capabilities of the 705 and the statement of the problem. The programmer can then break the problem down into a list of detailed instructions which the computer can follow.

The IBM 705 is a general purpose system. It can obey a number of basic commands but it is not mechanized to execute these commands in any particular sequence other than the order in which they occur in memory. The programmer arranges the sequence before instructions are stored or he inserts special instructions to alter the sequence in which the program is to be followed. In this way, a general purpose machine can be adapted for work on a great variety of problems. The change from one procedure to another merely involves storing the appropriate program and feeding the required data.

The programmer, therefore, is primarily interested in two things. One is the list of all different instructions that can be performed, and precisely what each instruction accomplishes. The other is memory, the main storage for all data.

Address System

An IBM 705 memory can store as many as 80,000 characters. To make information available, there must be some system or order to storage. The programmer, for example, must know this order if he is to keep track of where data and instructions are to be located. An incoming record may contain 500 characters; therefore, some 500 positions of storage must be assigned to receive this input record. Other areas may be needed for output records, constants, working space for intermediate results, instructions, and so on.

Each memory position is assigned a number or address, beginning with 00000 and continuing to 19,999, 39,999, or 79,999, depending upon the model of system being used.

One method of visualizing the storage of data is to compare memory with a hat check room with capacity to store a certain number of hats (Figure 16). Instead of hats, the memory stores characters. Each hat is located in a separate bin, just as each character is in a separate position of core memory. The hat bins are all arranged in a regular order with a number for each one. In memory, each position is assigned an address.

When the programmer arranges information in memory, he decides where each piece of information will go. That is, he decides what addresses will be used to refer to the data. When the data are to be

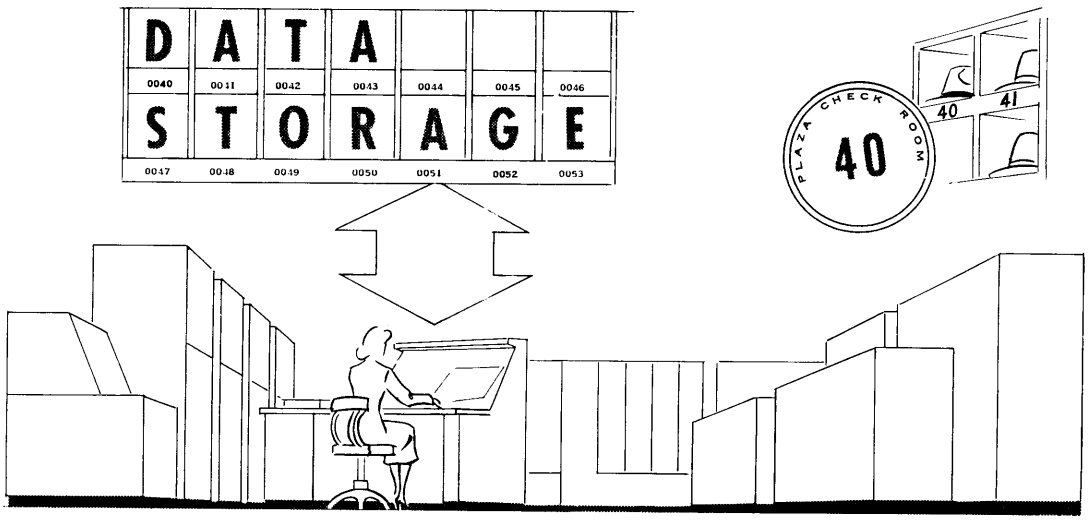


Figure 16. Memory Address System

transferred, the actual data will not be specified but only the address. This method may also be compared to the hat check room. When a hat is called for, it is not the hat itself that is described, but its location or address as shown on the check ticket.

The address system of the 705 includes not only memory, but all of the other components as well. Each input-output unit is also assigned a code number or address. It can then be called upon by specific instructions to read, write, backspace, eject a form, and so on. The addresses for these units are explained under the sections describing input-output components.

Four character positions are provided in the address part of an instruction for addressing memory. Since any memory address of 10,000 or above is actually a five-digit number, a system of zone coding is used to keep within the four-character restriction. Zone coding is placed over the first and last position of a memory address to represent the high-order digits 1 through 7.

B and A zone bits over the high-order position of the instruction address have a decimal value of 2 and 1, respectively. The B zone bit over the low-order position of the instruction address has a decimal value of 4. By adding the decimal values of the zone bits in various combinations, it is possible to represent all the necessary decimal digits used in the fifth-order position of an actual memory address. For example, the first 10,000 positions of memory, 00000 to 09999, have a zero value in the fifth-order position; therefore, no zone bits are required in the high- or low-order positions of the instruction address. Memory positions 20,000 to 29,999 have a high-order address value of 2; therefore, a B bit over the high-order position of the instruction address is required. Memory addresses referring to the upper 40,000 positions of a memory always have a B bit in the low-order position, plus the proper B and A bits in the high-order position of the instruction memory address. A chart of actual memory addresses and coded memory addresses follows:

ACTUAL MEMORY ADDRESS	CODED MEMORY ADDRESS
00000 through 09999	$\overset{00}{0}\,0\,0\,0\,\overset{00}{0}$ through $\overset{00}{9}\,9\,9\,9\,\overset{00}{9}$
10000 through 19999	$\overset{01}{0}\,0\,0\,0\,\overset{00}{0}$ through $\overset{01}{9}\,9\,9\,9\,\overset{00}{9}$
20000 through 29999	$\overset{10}{0}\,0\,0\,0\,\overset{00}{0}$ through $\overset{10}{9}\,9\,9\,9\,\overset{00}{9}$
30000 through 39999	$\overset{11}{0}\,0\,0\,0\,\overset{00}{0}$ through $\overset{11}{9}\,9\,9\,9\,\overset{00}{9}$
40000 through 49999	$\overset{00}{0}\,0\,0\,0\,\overset{10}{0}$ through $\overset{00}{9}\,9\,9\,9\,\overset{10}{9}$
50000 through 59999	$\overset{01}{0}\,0\,0\,0\,\overset{10}{0}$ through $\overset{01}{9}\,9\,9\,9\,\overset{10}{9}$
60000 through 69999	$\overset{10}{0}\,0\,0\,0\,\overset{10}{0}$ through $\overset{10}{9}\,9\,9\,9\,\overset{10}{9}$
70000 through 79999	$\overset{11}{0}\,0\,0\,0\,\overset{10}{0}$ through $\overset{11}{9}\,9\,9\,9\,\overset{10}{9}$

Accumulator and Auxiliary Storage

A number of small magnetic core storage units are also provided in the IBM 705. These units temporarily store information from main memory, accumulate the results of arithmetic, and perform various other functions of data handling. They serve as counters and character registers, but they are also used in operations such as shifting, comparison, sign manipulation, arranging a record for printing, and controlling the movement of information from one section of memory to another. They are only accessible through memory; they cannot receive or transmit data directly to any input or output device.

Storage is divided into two sections: an accumulator of 256-character capacity, and 15 auxiliary storage units (Figure 17). Each auxiliary unit has a capacity of 16 characters, except number 15 which has a capacity of 32.

A counter or storage register normally accommodates a field or quantity made up of a fixed number of characters. A 10-digit counter, for example, always stores the units digit in its right-hand position, the tens in the next position, hundreds in the next, and so on, to the capacity of the counter. Whenever the counter is cleared or reset, all ten positions are read out. Left zeros may or may not be used, depending upon the function of the counter.

The accumulator differs considerably from the normal concept of a counter. Not only can it store any 705 character, including special symbols, but the location

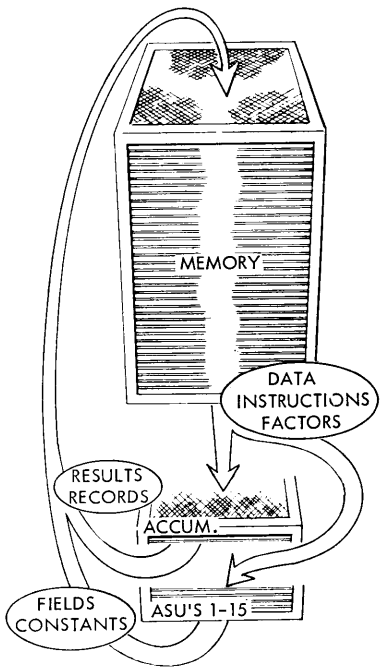


Figure 17. Memory, Accumulator, and Auxiliary Storage

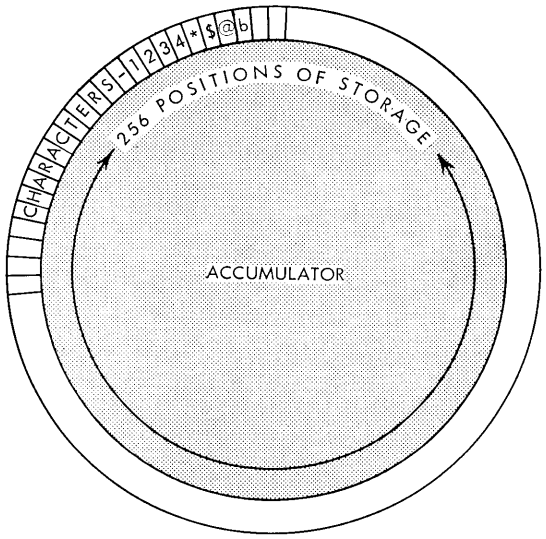


Figure 18. Accumulator Storage

of the stored field within the 256 possible positions is not fixed. And, when the field or result is read out, it is regenerated in the accumulator in the same manner as it would be in memory.

The accumulator can probably be illustrated best by assuming that storage is in the form of a circle. The 256 positions may then be assumed to be arranged around the circumference of the circle (Figure 18). Stored data may occupy any of these positions. But, since the entire capacity may not be needed, the field or record in storage must be limited or defined when it is to be used. For example, a six-character field would occupy only six spaces, a six-position segment of the entire circle (Figure 19).

When the field is placed in the accumulator, its units position is automatically placed at the location of an indexing device called the starting point counter. The counter remains in this position until moved by instructions in the program or until multiplication or division is performed. Moving the counter to the right (by instructions) has the effect of lengthening the field and adding zeros, while moving the counter to the left shortens the field (Figure 20).

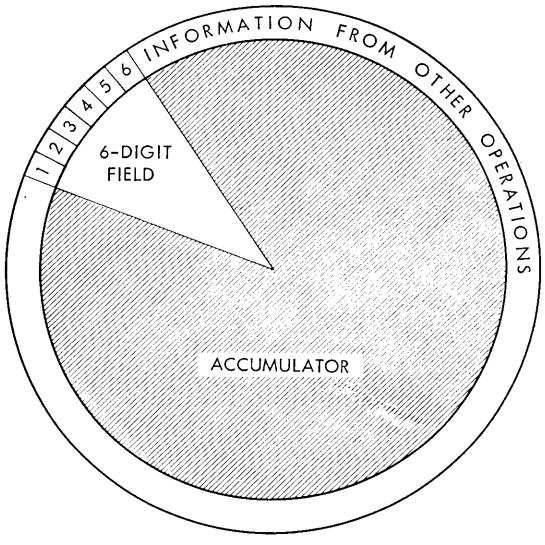


Figure 19. Field in Accumulator

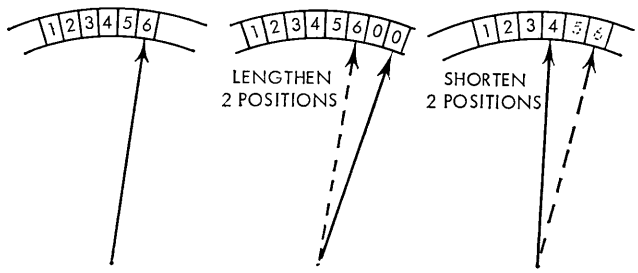


Figure 20. Starting Point Counter

The left-hand, or high-order, position of the accumulator field is limited by a special character, a storage mark. The mark is automatically placed in the storage position next to the high-order character. It may also be moved by instruction. This enlarges the field when moved to the left (again, zeros are automatically added), or cuts off part of the field when moved to the right (Figure 21). In the text and illustrations, the mark is represented by the letter "a." It is normally stored by the 705 only in accumulator or auxiliary storage as 0 00 0000.

In use, the accumulator readily accommodates any field or record up to 255 characters in length. One position is occupied by the storage mark to define the left-hand limit of the stored data. When information is returned to memory from the accumulator, only the characters bounded by the starting point counter and the storage mark are read out. Although the accumulator may also contain other information from previous operations, this is ignored.

Once information is placed in storage, it remains there until replaced by storing other data or the result of calculation. A field can therefore be placed in storage and then be "read out" to any number of locations in main memory. Each time a transfer of data to memory occurs, the data are automatically restored in the accumulator (Figure 22).

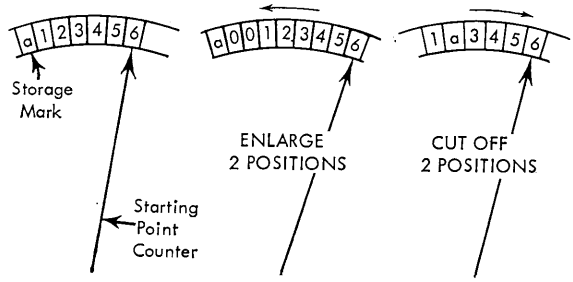


Figure 21. Storage Mark

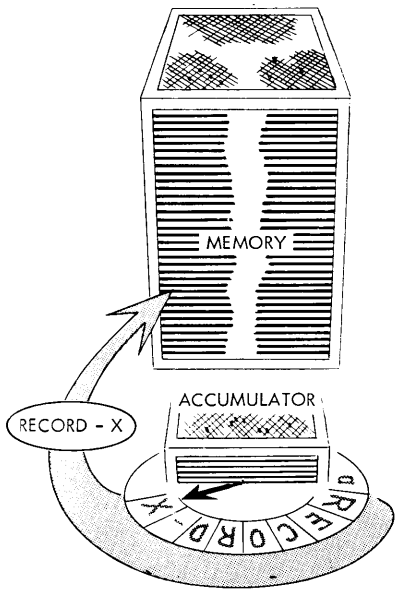


Figure 22. Data Flow from Accumulator to Memory

The accumulator can store the results of any arithmetic operation: addition, subtraction, multiplication, or division. However, a computed product or quotient cannot be larger than 128 digits. One factor of a calculation is always stored in memory; the other is placed in the accumulator. For example, to add field *A* to field *B*, either *A* or *B* may be placed in storage. The actual operation of adding is performed by other machine circuitry (arithmetic and logical unit). The sum of the two fields replaces the original field in the accumulator (Figure 23).

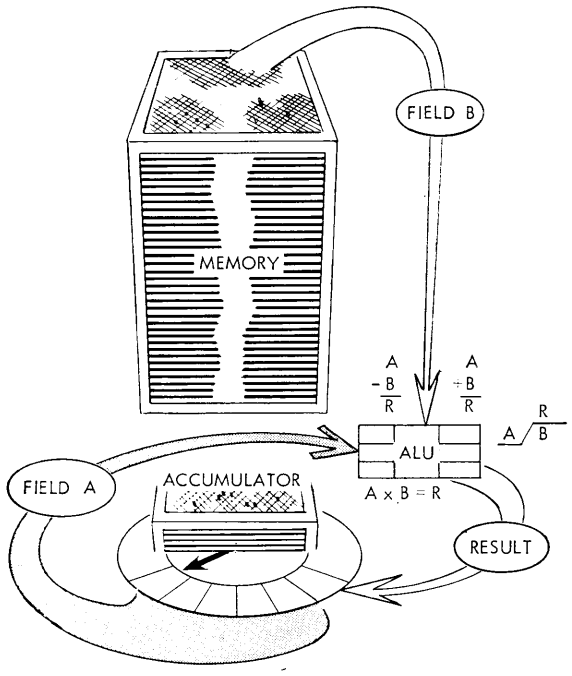


Figure 23. Accumulator in Arithmetic Operations

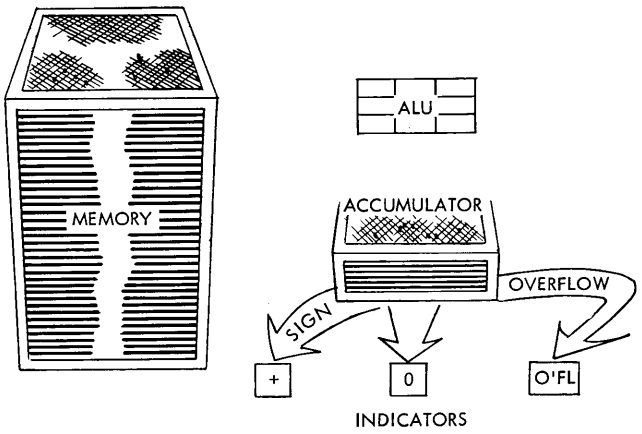


Figure 24. Plus, Zero, and Overflow Indicators

Both positive and negative results are stored as true numbers. A sign indicator automatically registers the correct algebraic sign of a result by being turned on for plus, off for negative. A second indicator is used to register that the contents of the accumulator equal 0. Provision is also made to recognize overflow (Figure 24). For example, if the sum of $A + B$ contains more digits than either *A* or *B*, an overflow check indicator is turned on. This prevents the return to memory of a sum that exceeds the capacity of the space reserved for it. In this way, the overflow indicator can be used to flag results that are greater than predetermined limits. (The indicator also signals the violation of certain rules that must be followed during a division operation.)

Other uses of accumulator storage are more fully explained in later sections.

Auxiliary storage can also be represented as being circular. And, as in the accumulator, a total of 256 character storage positions are available around the circumference of the circle. But, unlike the accumulator, the circle is divided into segments by 15 preset starting point counters (Figure 25). In use, this makes as many as 15 separate units of storage accessible to or from memory. Since the counters are fixed (they cannot be moved by instructions), each auxiliary storage unit (ASU) has a capacity of 16 positions, except number 15 which has a capacity of 32 positions.

When information is placed in an ASU from memory, the right-hand character is always stored at the position of its starting point counter. The left-hand limit is defined by a storage mark in the same way as in the accumulator (Figure 26). A mark in any unit may be moved in either direction around the circle by instructions.

If information placed in storage from memory extends beyond the capacity of a particular unit, it is automatically stored in adjacent positions in the next

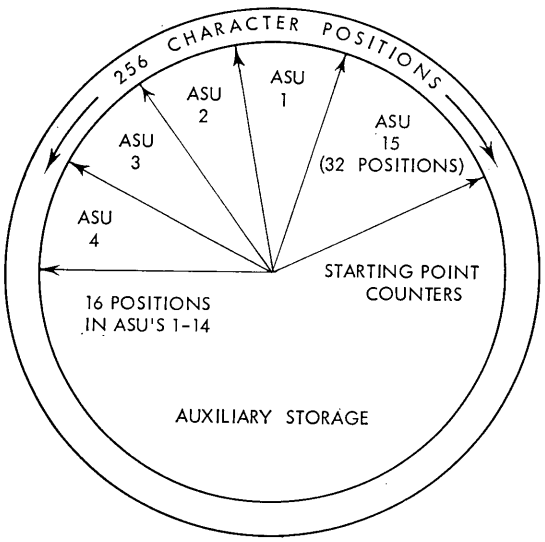


Figure 25. Auxiliary Storage Units

unit, counterclockwise around the circle (Figure 27). In this way, auxiliary storage can either be used to its full capacity of 256 positions (one position is occupied by a storage mark) or be used in segments of one or more ASU's. Data returned to memory are defined by the starting point counter of the unit specified and the next left storage mark.

In general, the functions of ASU's are the same as those of the accumulator. They provide convenient and efficient storage for small fields and constants which may be used a number of times in the same problem. They can also control movement of data in memory, store results of addition and subtraction, and arrange records for printing. However, multiplication and division use the accumulator only. These operations cannot be performed with auxiliary storage.

Two indicators are provided for auxiliary storage to register automatically the proper algebraic sign of results and to indicate zero balances. The same two indicators serve all of auxiliary storage. Therefore, they show the sign and zero condition of the last ASU used.

The overflow indicator shows an overflow condition in either accumulator or auxiliary storage.

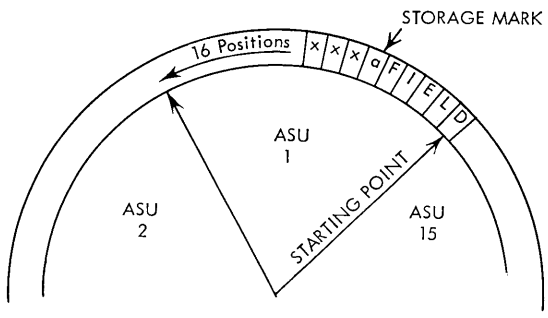


Figure 26. Record Definition, Auxiliary Storage

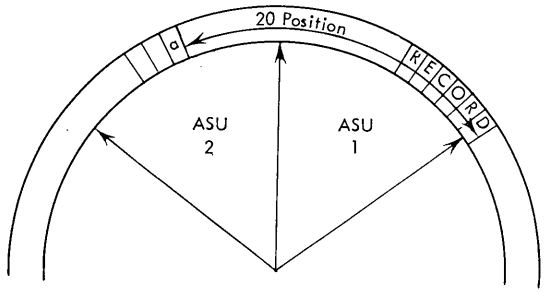


Figure 27. Coupling ASU's

Comparison

The contents of either accumulator or auxiliary storage can be "compared" against information in memory. Fields, records, quantities, or instructions in storage can be determined to be higher, lower, or equal to corresponding data in memory.

The results of a comparison may be determined by testing for a high, equal, or low condition. When the storage contents are higher than memory, an instruction called "transfer on high" can cause the machine to alter the sequence of instructions that it executes in a predetermined manner. An instruction called "transfer on equal" performs the same function when the results of a comparison are equal. If the comparison is low, the normal sequence of instructions is executed.

In this manner, the machine can be directed by instructions to change its method of processing as a result of whether one record matches, is higher than, or is lower than another.

The storage contents are found to be higher, lower, or equal to data in memory according to the preset sequence of 705 characters. This sequence is: blank . ☐ ≠ & $ * − / , % # @ 0̇ A through I 0̄ J through R ≠ S through Z 0 through 9. This is also the same order established for IBM punched card machines, so that records arranged in sequence by the 705 conforms to the standard sequence of card files.

For example, if the name Jones in storage is compared with the name Brown in memory, Jones is "higher." Compared with Smith, Jones is "lower." By using a series of comparisons, a number of records in memory can be arranged in alphabetic sequence, or input or output records can be checked for sequence as part of a processing procedure.

Comparisons of numerical fields can be made either by using direct comparisons or by using arithmetic. After subtraction of the storage field from a memory field, a negative result indicates that the storage field

is higher. A positive result indicates that the storage field is lower (plus-indicator on), while a zero result indicates that the fields are equal (zero-indicator on).

All indications can be tested or "interrogated" by instructions. Figure 28 is a schematic of data flow with use of the zero indicator.

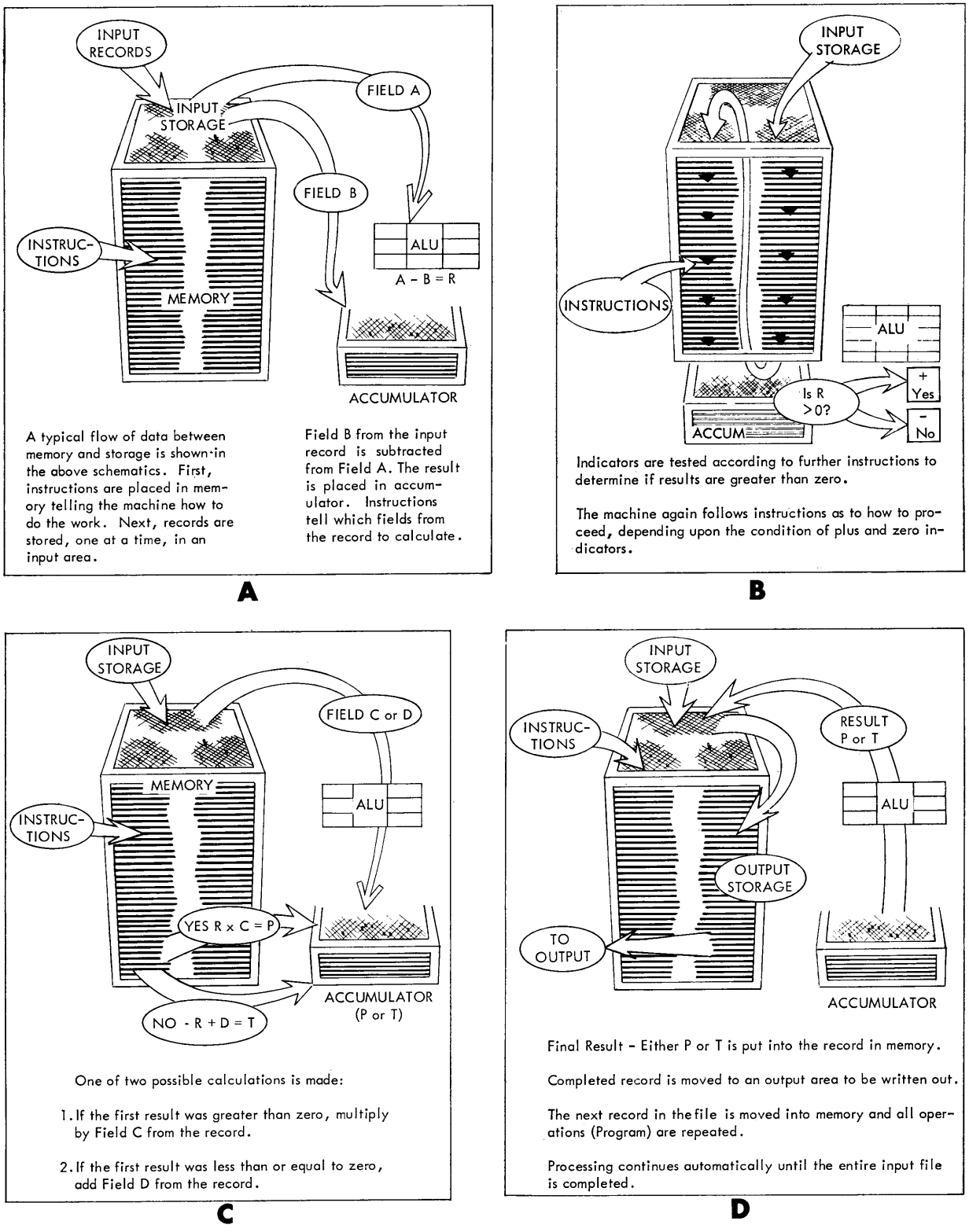


Figure 28. Schematic, Data Flow between Memory, Accumulator, and ALU

Magnetic Drum Storage

Commercial data processing procedures are well known for their complexity of detail. A relatively simple accounting job often requires several hundred possible operations, even more when all possible exceptions to normal routines are considered. For example, one IBM plant payroll takes some 6200 different 705 instructions to cover all variations in the payroll, tax, deduction and employee benefit calculations. Of these instructions, an average of only 750 are used to calculate the complete data for any one employee.

Also complete data handling usually involves a considerable amount of reference information which must be related to current transactions in some manner. Such reference material may be in the form of tables, rate schedules, results from previous calculations, constants, code lists, and so on. Public utility billing is a good example of this. Electric power is normally priced at a graduated rate taken from a pre-established rate schedule. A factory, using large amounts of current, may be charged a substantially cheaper rate than an average home owner. The home owner's rate may also vary, depending on his type of service, or whether he owns a water heater, stove, or other electric appliances.

The application of a computer to such problems is more efficient if enough capacity is available to store all the necessary data in core memory. However, even with the 80,000 positions of storage offered in the largest model 705, memory may become filled with instructions and input-output record areas, leaving insufficient space for reference data. Many applications have a definite need for supplemental storage, accessible at relatively high speed, but cheaper than core storage. The magnetic drum, also available with the 705, is designed to fill this need.

The 705 drum is a steel cylinder enclosed in a copper sleeve. It is 10.7 inches in diameter by 12.5 inches in length (Figure 29). The copper surface is plated with a cobalt and nickel alloy similar to the metal used in alnico magnets. The coating on the surface of the drum is the actual magnetic storage medium.

If an area of this material is placed in a magnetic field, it becomes magnetized. And after the magnetizing force is removed, the magnetism is retained indefinitely. The area affected can be quite small (on the drum it is about .017 inches in length) 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 (about 3730 rpm), 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 is inserted at the ends of the core, providing a magnetic gap. The head assembly is positioned close to the drum in such a way that magnetic lines of force produced by the write head "fringe" around the gap and flow through the alloy surface (Figure 30).

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 the binary coding of all 705 characters. Because the drum is rotating while writing takes place, an extremely short duration of the write current limits the area magnetized. In this way, the size of the cell is almost the same as if the drum were motionless.

When a cell that has been previously 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. Also, reading is not destructive because the condition of a cell is not changed as it passes the head. Unlike core memory, no regeneration process is needed 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

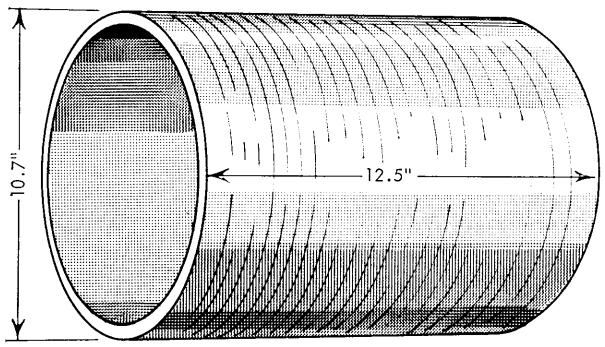


Figure 29. Magnetic Drum

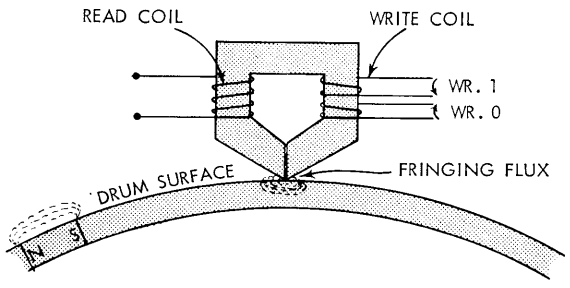


Figure 30. Drum Recording

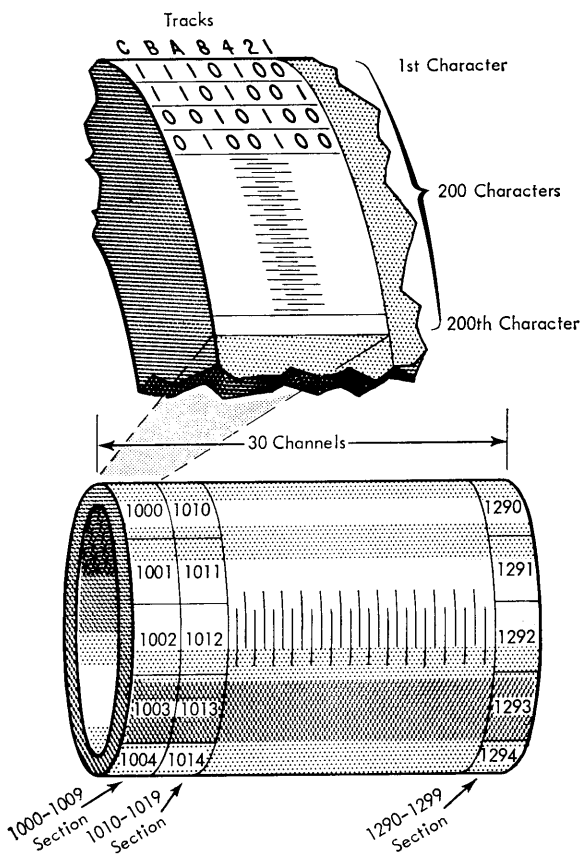


Figure 31. Reading and Writing on the Drum

the 705 system is turned off. Information is replaced only when new information is written.

The drum uses 210 read-write heads that read or write information in 210 tracks around its circumference (Figure 31). Tracks are grouped into 30 channels of seven tracks, each of which conforms to the seven-bit binary coding structure of 705 characters. One channel in a track would contain all C bits; the second, B bits; the third, A, and so on. A channel has a capacity of 2,000 characters and is divided into ten sections of 200 characters each.

Any section of the drum is accessible to memory, but individual characters within a section are not. Since data consisting of a variable number of characters may be stored on the drum, reading or writing begins at the first position of a section and continues automatically into following higher-order sections. The end of a block of information is limited by a special character called a "drum mark." The character is automatically written at the end of a block of data. It stops entry into memory when reading.

A record or block of data may be of any length within the capacity of the entire drum, or it may be as short as one character. Drum storage may contain a number of blocks of data, each beginning at the first position of a section and extending through one or more following sections, then ending with a drum mark.

Because reading or writing can occur only when the specified section is passing the heads, the access time to a section may vary, depending upon the distance to be traveled by that section to the head. The average time to locate the first character is eight milliseconds. Thereafter, characters can be read or written consecutively at the rate of .04 millisecond per character.

Figure 32 shows schematically the flow of data between the drum and memory.

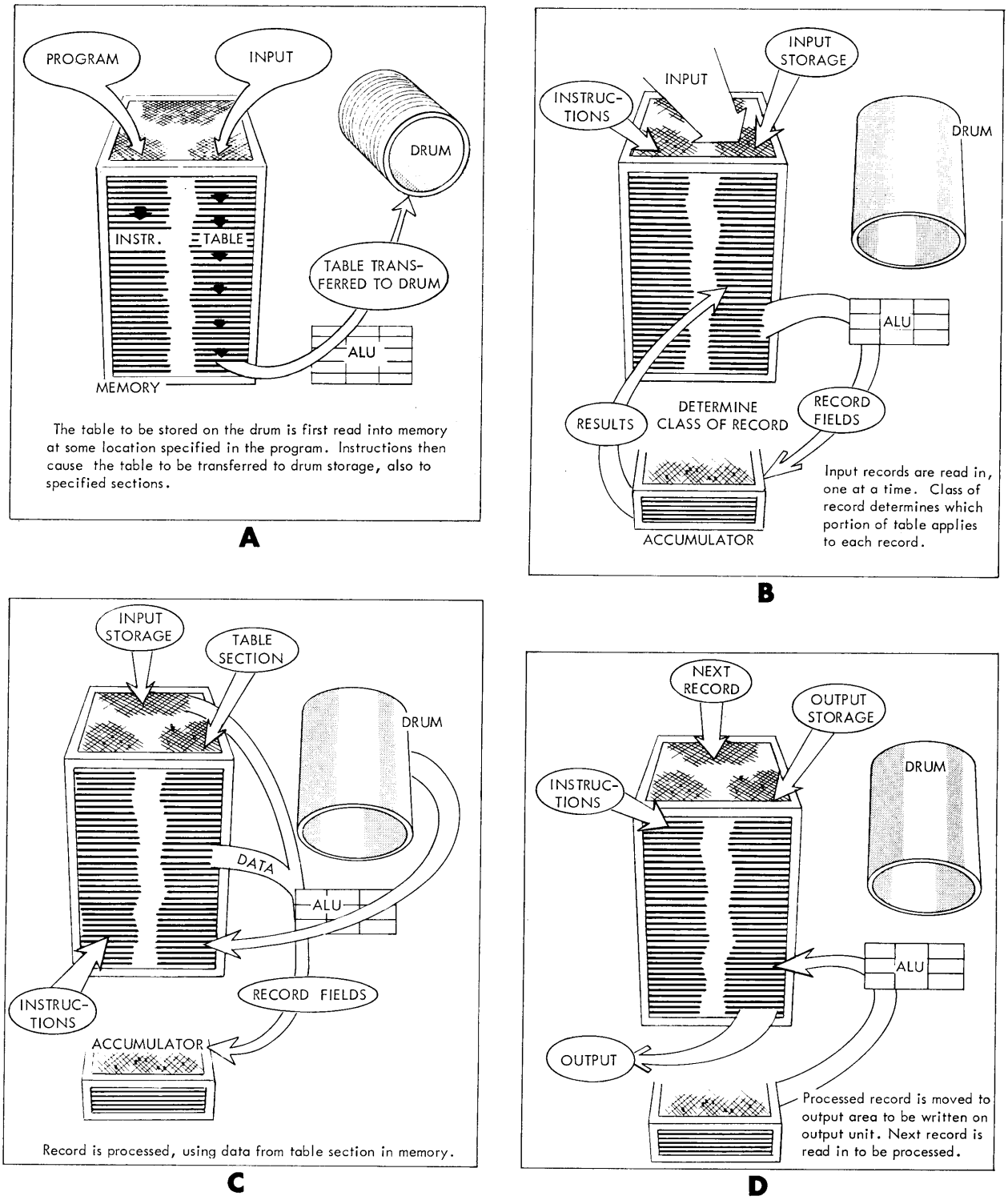


Figure 32. Schematic, Data Flow Using the Drum for Additional Storage

The history of magnetic recording dates back to 1898 when the Danish scientist, Valdemar Poulsen, experimented with recording sound on a steel wire. At that time, the performance of his machine was severely handicapped for lack of proper amplifiers and a uniform grade of wire.

The design of magnetic recorders was greatly advanced in Germany during World War II where equipment capable of a high degree of fidelity was discovered during the American occupation. Further developments led to the use of magnetic tape, rather than wire, for nearly all professional and commercial sound recording equipment.

Because tape is light, compact, and durable, it provides an ideal record handling and file storage medium for high-speed data processing equipment. Instead of recording sound, the computer records data in coded form.

Magnetic tape also possesses a unique characteristic found in no other type of record— that of automatic erasure. Although recording is permanent, affecting the magnetic state of the tape surface itself, any previous recording is destroyed by the writing operation. This means that tape can be used again and again with significant saving in recording costs.

IBM magnetic tape used by the 705 is similar to the tape used in home recording devices. One surface is coated with a layer of iron oxide particles — so small that over one trillion such particles cover only one-quarter square inch of area. The coating, with suitable binder, forms a layer about .0006 inch thick on the plastic base. Recording actually occurs in this oxide layer (Figure 33).

The plastic base is one-half inch wide and may be purchased as either acetate or Mylar®. Somewhat more

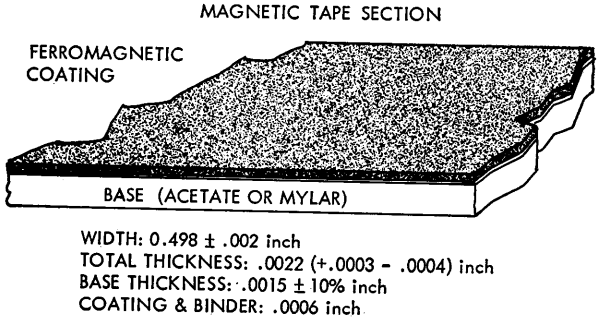


Figure 33. Section of Magnetic Tape

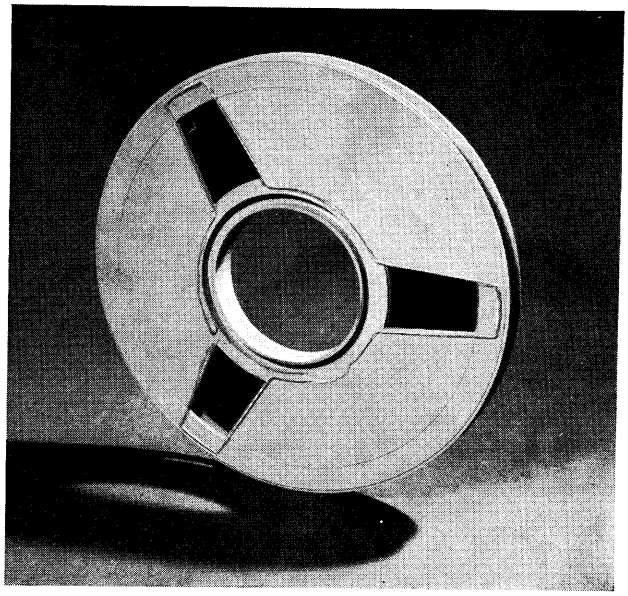


Figure 34. Magnetic Tape Reel

expensive, Mylar is a more stable plastic material under changing atmospheric conditions and exhibits approximately fifty percent more tensile strength than acetate. IBM Durexcel magnetic tape is also available. It combines outstanding durability and reliability through an advanced development in the formulation of magnetic tape coating. This produces extremely long error-free life for the tape.

A maximum of 2400 feet of tape may be wound on a plastic reel 10½ inches in diameter. Shorter lengths of as few as 50 feet may also be used, if convenient (Figure 34).

The process of recording is similar to that previously described for drum storage. Information is written by changing the status of magnetization of tiny spots in the oxide coating on the tape from positive to negative or from negative to positive (Figure 35). All data are coded in 705 character code. A change in magnetization represents a 1; the lack of change, a 0. Because seven combinations of 1's and 0's are needed,

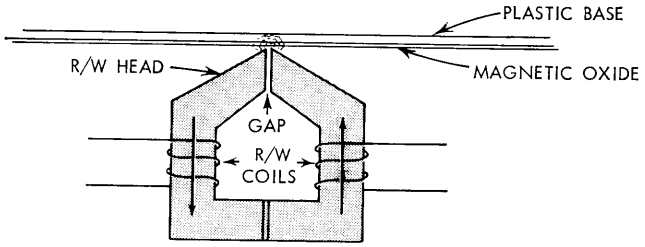


Figure 35. Read-Write Coils and Magnetic Tape

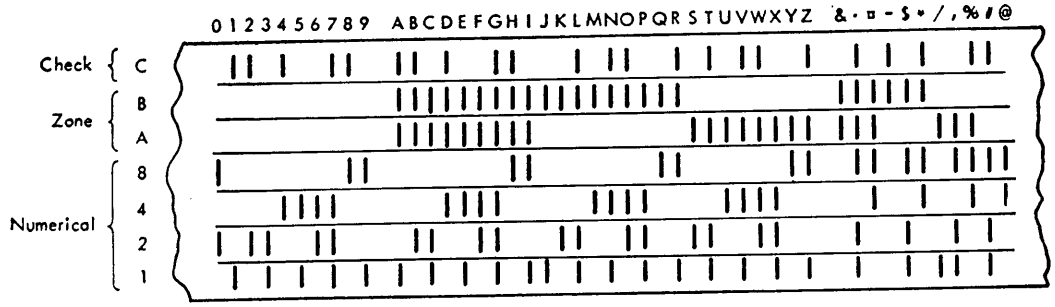


Figure 36. Tape Character Coding

writing is positioned in seven tracks horizontally along the length of the tape. These tracks correspond to the seven-bit binary coding structure of 705 characters (Figure 36).

Reading is accomplished by sensing the changes in magnetization of the oxide as "bits" or 1's as the tape is passed over the reading heads of the tape unit.

Tape Records

Conventional recording equipment of all types normally handles records of fixed length. The punched card user, for example, is only too well aware of the limitation of cards to a maximum length of 51, 60, 66, or 80 columns. This means that the ability to record information concerning one particular item or fact is restricted to the capacity of a single card. If more information is needed, two or more cards are required. However, the recording capacity is not doubled, since the identifying data, such as name and part number, must also be punched in all cards making up the multiple record.

A 705 tape record is not restricted to any fixed length of characters, fields, words or blocks. Records may be of any practical size within the available capacity of the memory area assigned to the storage of data. This feature not only allows for the writing of all information pertinent to an item in a single continuous record, but also eliminates the need for repeating the identification in multiple records. Thus, as much information as is needed can be conveniently included in its most compact form (Figure 37).

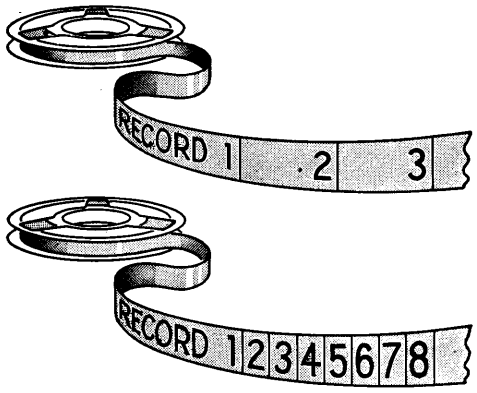


Figure 37. Schematic, Variable Record Length

Records are separated on tape by a record gap – a prescribed length of blank tape, approximately ¾ inch in length. During writing, the gap is automatically produced at the end of the 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 provides a space to allow 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 38).

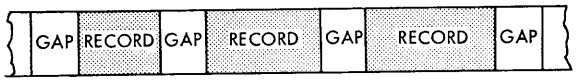


Figure 38. Tape Records Shown with Inter-record Gap

This method of recording also provides capability for handling records of variable length within the same file. Consequently, only pertinent data need be recorded – no unused "columns" or word blocks are necessary to fill out or pack the record to conform to any fixed length. Reading of tape begins with the first character following a gap and is automatically terminated *only* when the next gap is sensed.

If short records are handled, more than one record may be written between gaps. In this case, individual records are separated by a special character called a "record mark." The mark is shown as the symbol ǂ in the text and illustrations (Figure 39). It is also read into memory in the same manner as any other character where it is used to define separate records for processing. The device is mainly useful to permit read-

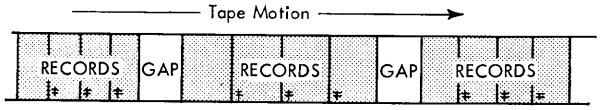


Figure 39. Grouped Records Defined by Record Marks

ing and writing large blocks of data without any delay in starting and stopping the tape for each individual record.

An inter-record gap, followed by a special single character record, is used to mark the end of a file. The character, a "tape mark," is emitted from the tape control at the time the file is written. Normally, it never appears in memory, but only on tape. It is represented in the text and illustrations as TM (Figure 40).

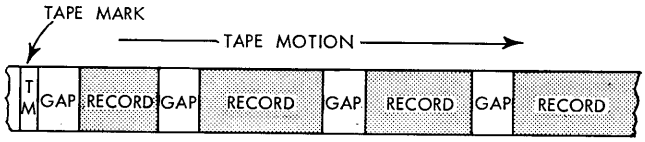


Figure 40. Tape Mark at End of File

Sensing a tape mark automatically turns on an input-output indicator in the tape unit signaling that no more records are to be read from this reel. Normally, instructions should then rewind the tape and halt the machine, since all the records in the file are assumed to be processed. Depressing the unload key on the tape unit then turns off the indicator and the reel may be removed from the unit.

However, more than one file of records may be written on a single reel. In this case, the indicator may be turned off by instructions, and reading of the next file of records can continue. Or, sensing the tape mark may only indicate the end of one of a number of reels that make up the complete file. Instructions may then be given to rewind and continue reading from the next reel which may be ready on a second tape unit. In this way, operation of the system is not interrupted while a reel change is being made.

Like movie film, the tape must have some blank space at the beginning and end of the reel. This blank end can be threaded through the feeding mechanism of the tape unit. Reflective spots of aluminum foil, placed on the tape by the operator at any desired distance from the ends of the reel, are photoelectrically sensed to indicate the physical end of tape and the starting point for recording (Figure 41).

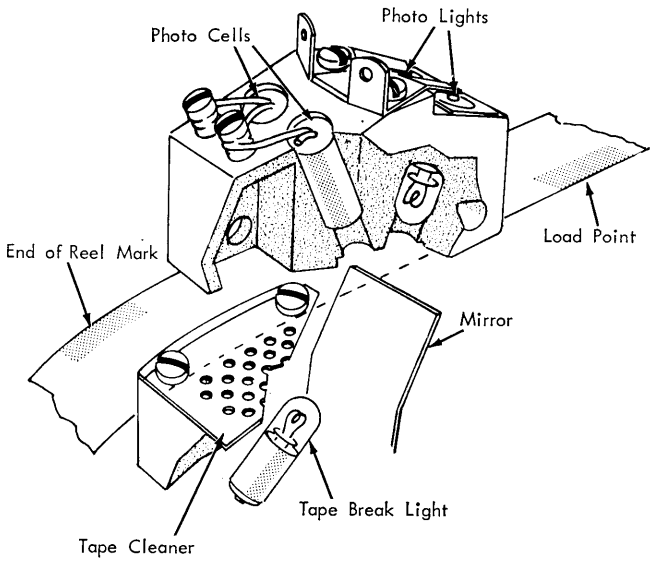


Figure 41. Photo-Cell Sensing, IBM 727 Magnetic Tape Unit

The tape may also be backspaced under control of instructions from the 705. Tape motion is reversed until the previous inter-record gap is sensed. This feature provides for rereading or rewriting one record or block of data when an error condition is sensed. However, the tape may be backspaced any number of records desired under repeated instructions.

During operation, if an error condition is detected, the machine can be instructed to backspace and "try again" to correct the mistake. The number of attempts to read or write can be limited to any number, normally two or three. If the error cannot be corrected automatically, instructions can stop the machine and indicate to the operator where the failure has occurred.

Because the writing operation automatically erases any previous information on the tape, a file protection device is provided to prevent accidental erasure of information to be saved for further reference. A circular groove is molded around the center of each reel to fit a demountable plastic ring. Without the ring in place, writing is suppressed and only reading can occur. The file in this condition is protected. When the ring is in place, either reading or writing can occur (Figure 42).

During writing operation, the reflective spot signals end of reel to the tape unit. A tape mark can then be emitted by a write tape mark instruction to signify end of file when reading. Sensing the spot also turns on the tape unit input-output indicator.

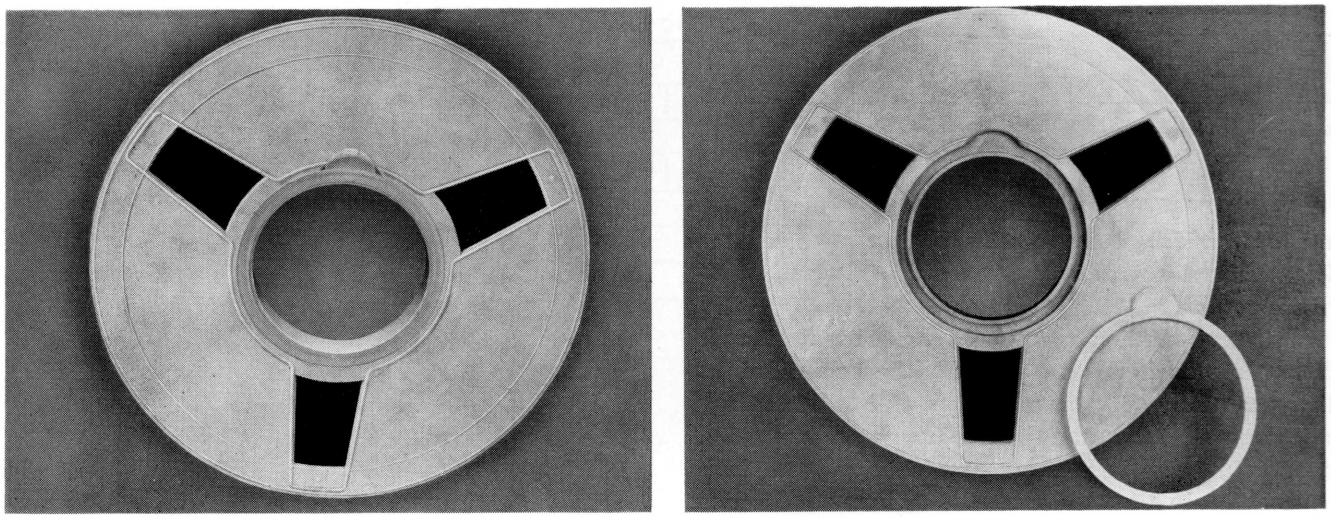


Figure 42. File Protection Device

Tape Unit

A 705 system can be equipped with any practical number of tape units for reading or writing magnetic tape. The same unit performs either operation. Figure 43 shows schematically the position of the tape reels in relation to the read-write heads and feed rollers of the IBM 727 Magnetic Tape Unit. During reading or writing, tape is transported from the file (left) reel past the heads to the machine (right) reel. During rewind, the tape motion is reversed.

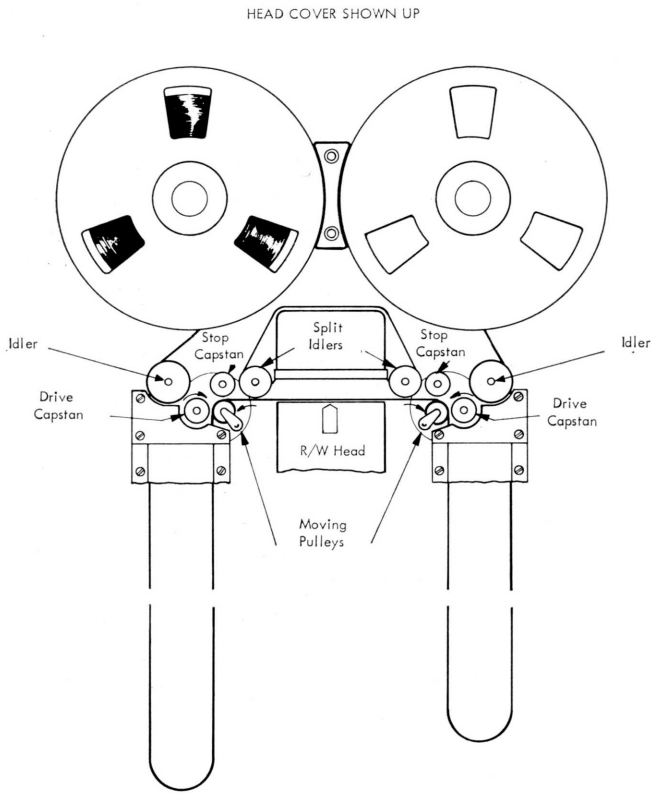


Figure 43. Path of Tape through Machine

Since it is impossible to start and stop high-speed motion under control of the reels above, a loop of tape is drawn into vacuum columns before it passes over the feed rollers. As tape is drawn from one column, it is replenished from the reel above. As it is fed into the opposite column, the associated reel takes up slack tape.

The head assembly is built on two vertical plates, the lower of which is stationary. The upper plate may be moved up and down under control of the load and unload keys on the front of the unit or be automatically raised during a high-speed rewind. Movement is about one inch away from the lower assembly to allow threading of tape by the operator. Changing reels and threading of tape takes approximately one or two minutes.

Rewind is under control of an instruction from the 705 or may be started manually by depressing the load-rewind key. During rewind, the tape is transported from the machine reel back to the file reel. If the tape has been wound onto the machine reel to a thickness of at least ½ inch, the unit goes into a high speed rewind. The read-write head is raised automatically; tape is pulled from the vacuum columns and passed directly over the feed rollers at an average speed of 500 inches per second. When less than one-half inch thickness of tape is on the machine reel, rewind is executed at the normal speed of 75 inches per second. A full reel can be rewound in about 1.2 minutes.

During a high-speed rewind, tape is passed between a light source located on the lower head plate and the load-point photo cell located on the upper head plate.

Figure 44. IBM 727 Magnetic Tape Unit for Use on IBM 705 I and II

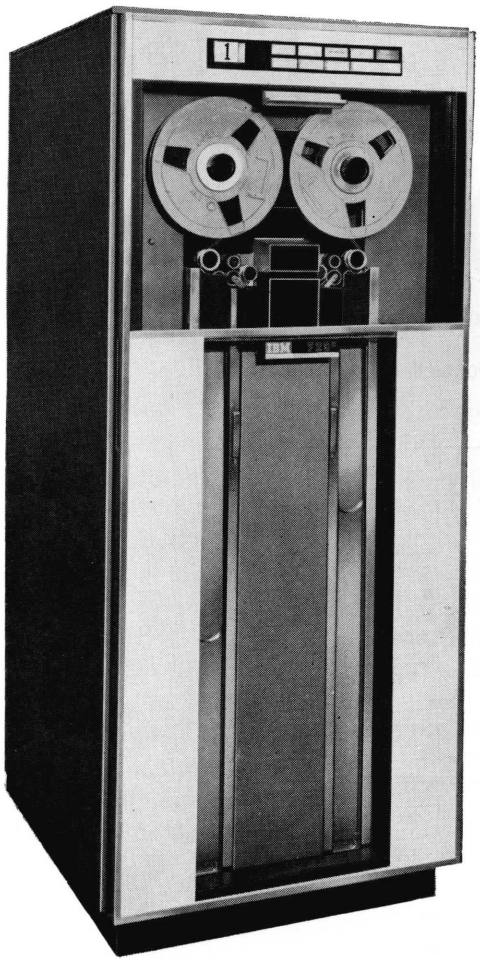


Figure 45. IBM 729 III Magnetic Tape Unit for Use on IBM 705 III

If the tape breaks, the light strikes the photo cell causing the tape unit to stop.

Tape units are available in three models: the 727 (Figure 44) for use with the 705 I and II, and the 729 III (Figure 45) and 729 I used with the 705 III. Both 729 models make use of improved methods of checking the accuracy of reading and writing. These features are more fully explained in following chapters. The 729 III is transistorized.

Each unit is furnished with a selector dial which may be set by the operator from 0-9. The selected number then becomes the "address" of the unit. Instructions in the 705 can then call the unit into use by specifying this assigned address.

The 727 and 729 I write at a density of 200 characters to one inch of tape. This means that a 100-character record would occupy one-half inch of space; a 400-character record, two inches of tape, and so on. Tape is moved past the read-write head at a speed of 75 inches per second. The maximum rate of reading or writing is therefore 15,000 characters per second or 67 microseconds per character.

The 729 III writes at a density of 556 characters per inch. The 100-character record occupies slightly more than .18 inch; a 400 character record, approximately .72 inch of tape (Figure 46). In the 729 III, tape is moved at a speed of 112.5 inches per second, providing a maximum reading or writing speed of approximately 62,500 characters per second (16 microseconds per character). Figure 47 compares the file space in tape reels between the two densities.

Figure 48 is a table showing the number of records that can be stored on a 2400-foot reel. Note that the

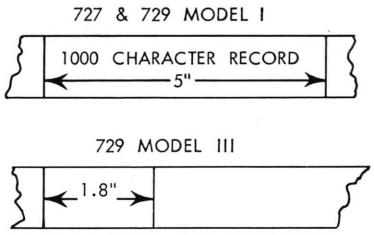


Figure 46. Comparison of Record Length

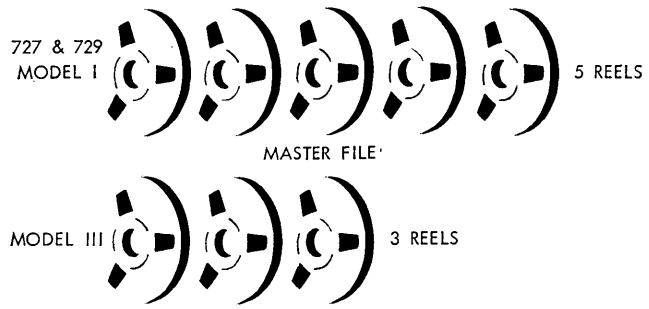


Figure 47. Comparison of File Storage

2370 Usable Feet Per Reel

Record Length	200 Char. Per Inch	556 Char. Per Inch
80 characters	24,730	31,810
100 characters	22,750	30,580
200 characters	16,250	25,620
300 characters	12,640	22,045
400 characters	10,340	19,345
500 characters	8,750	17,235
1000 characters	4,945	11,150
1500 characters	3,445	8,240
2000 characters	2,645	6,535

Figure 48. Tape Reel Capacity

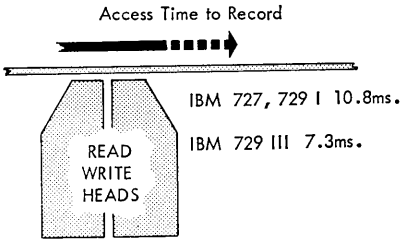


Figure 49. Tape Access Time

usable length of tape is stated as 2370 feet, allowing at least 15 feet of tape before and after the reflective spots.

Because an inter-record gap is spaced between each body of data on the tape, the total time to read a file must include this time to start and stop the tape between *each* block of information. This is termed "access time" – an interval required to accelerate the tape to proper processing speed. Access time averages 10.8 milliseconds on the 727 and 729 I, and 7.3 milliseconds on the 729 III (Figure 49). The following are comparisons of times for reading or writing records on the two tape units.

NO. OF RECORDS	NO. OF CHAR. PER RECORD	IBM 727, 729 I TIME (SECONDS)[1]	IBM 729 III TIME (SECONDS)[2]
100	80	1.62	.86
100	200	2.42	1.05
100	400	3.76	1.37
100	600	5.10	1.69
100	1000	7.78	2.33
100	2000	14.48	3.93

1. Time computed at 67 μs per character plus 10.8 milliseconds per record.
2. Time computed at 16 μs per character plus 7.3 milliseconds per record.

The previous table shows that the effective character rate increases in efficiency as longer records are processed – a further advantage of handling tape data. For example:

EFFECTIVE CHARACTER RATE PER 100 RECORDS

EFFECTIVE RATE (CHARACTERS PER SECOND)

CHAR. PER RECORD	727 AND 729 I	729 III
80	4,938	9,302
200	8,223	19,048
400	10,638	29,197
600	11,765	35,503
1000	12,853	42,918
2000	13,813	50,890

IBM 754 Tape Control

The operation of a tape unit is started by a "select" instruction from the 705 which specifies the code or address of the particular unit to be used. A following instruction specifies whether the unit is to read or write. If it is to read, the instruction specifies the location in memory where the data are to be placed. If it is to write, the location of the information already in memory is specified. Other instructions can call for backspacing, rewinding, writing a tape mark, and so on.

The 727 tape units are connected to the 705 through an IBM 754 Tape Control (Figure 50). This device receives or sends all tape data through an input-output cable connected directly to the central processing unit.

As many as ten 727 units, numbered from 0 to 9, can be connected to one 754. Two signal cables are attached to the control unit itself. One is used for odd-numbered tapes; the other, for even-numbered units. Only 727's with odd-numbered addresses may be connected to the odd cable; only those with even-numbered addresses can be connected to the even cable. Figure 51 is a schematic showing the proper arrangement of ten tape units attached to a 754, five on a side. As many as ten 754's may be connected to the 705 system.

The first two digits of a four-digit tape address always specify a tape operation. The third digit specifies one of ten possible 754's, while the units digit specifies one of ten possible tapes connected to that control unit. For example, the address 0231 specifies a tape operation using tape control 3 and tape unit 1.

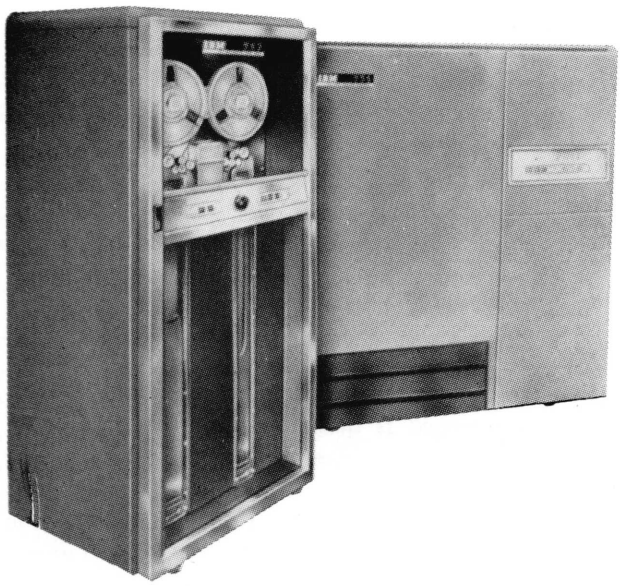


Figure 50. IBM 727 Magnetic Tape Unit and IBM 754 Tape Control

In the 705 system, two tapes may be operated at the same time, one for reading, the other for writing. Using this feature, the tape handling time for two files may be cut in half. A common example of this is the updating of master files. When transactions to a record have been applied, the next master record can be read into memory at the same time that the revised record is being written out on the new file.

When only one control unit is available for simultaneous reading and writing, the tape unit used to read must be assigned an even address. The unit used to write must have an odd address, or vice versa; that is, addresses 0201, 0203, 0205, and so on, may be assigned for reading; 0200, 0202, . . . 0208 may be used as addresses for writing. However, any two units may be selected during one procedure. One read-while-writing operation can use input tape 0201 and output 0202. The next can specify input 0208 and output 0205.

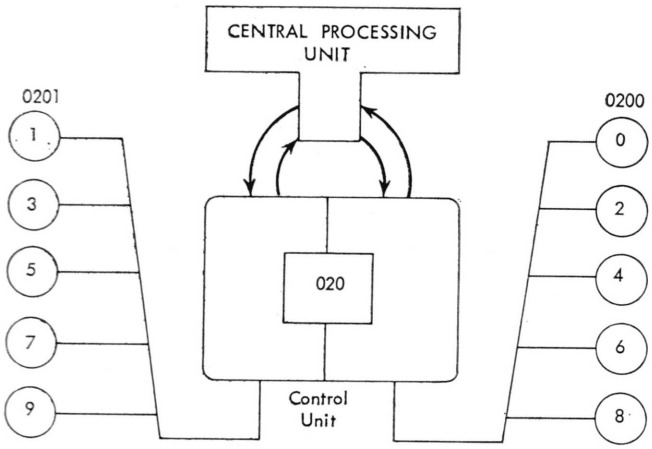


Figure 51. Tape Control, Read while Writing

When more than one control is available for input and output, such as 020x and 021x, there is no restriction on the assignment of tape addresses, if the units are connected to their corresponding odd or even cables.

During both reading and writing operations, validity checking of the records occurs in the tape control.

IBM 767 Data Synchronizer

The IBM 767 Data Synchronizer (Figure 52) is designed to operate as a tape control for the 729 I and III with the 705 III system. As many as five 729 I's and five 729 III's may be attached to one DS in any combination (Figure 53). If only 729 I's are used, a maximum of six DS's can be operated. If 729 III's are used, only four DS's may be operated. An IBM 748 Data Synchronizer Power Supply is used with each 767.

With 729 units connected to the system through two 767's, reading, writing and computing may occur simultaneously. This produces virtually independent operation of the tape units.

The 729 transmits data, character by character, to the 767. The data synchronizer stores the information

Figure 52. IBM 767 Data Synchronizer

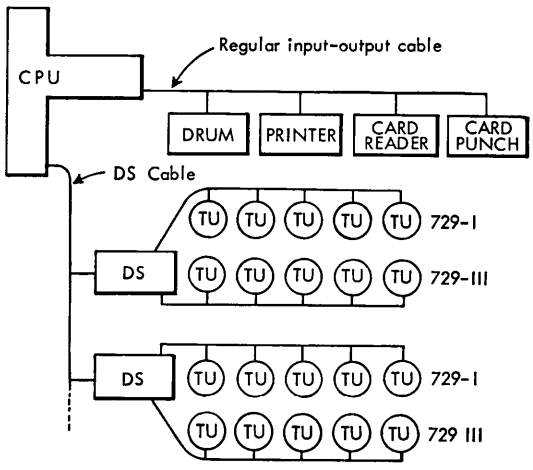


Figure 53. Data Synchronizer in an IBM 705 III System

from tape in two five-character position buffers or registers. As fast as the input buffer is filled, it transfers its contents to the output buffer and from there to memory.

When a particular tape unit is selected and instructed to read, the central processing unit proceeds to the next instruction without waiting for the tape to begin transmission of the data. For example, processing may continue on a previous record already in memory. While the CPU is processing other data, characters from the incoming record are read from tape, one at a time, and placed in the input buffer of the data synchronizer.

The capacity of the input buffer is five characters. As soon as it is filled, its contents are deposited, all at once, into the output buffer. The transfer is made in time to free the input buffer for the next five characters from the tape (Figure 54). The tape unit continues at full speed until the entire record is read.

As soon as the output buffer has received the five characters, it takes the next available machine cycle of the CPU to deposit its contents into memory. This step interrupts the CPU for a total of nine microseconds only. After this brief pause, the CPU returns

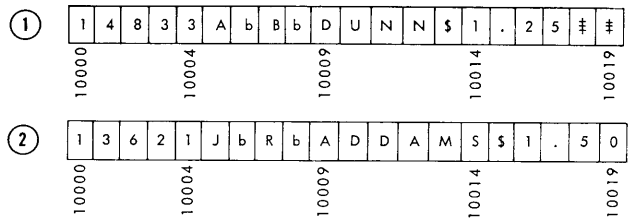


Figure 55. Placement of Group Marks by a DS

to processing once more until interrupted again by the DS with five more characters. Processing again continues with intermittent interruption until the tape record is completely transmitted to memory.

While the record is being read in this fashion, a synchronizer address counter controls the location in memory where the data are to be placed. It does this by starting at the memory location specified by the read instruction and stepping up by five as each group of characters is read in.

If the number of characters in the record is not an even multiple of five, the record is automatically extended while in the input buffer with from one to four special characters called group marks. If the number of characters in the record is an even multiple of five, no group marks are generated (Figure 55). The group mark character is later used to terminate a writing operation when the record is transmitted from memory to an output unit. It is represented in the illustration by the symbol ‡.

Once the entire record has been read into memory and the DS is selected, an indicator is set, showing that reading has been completed and that the data synchronizer is available for further use. The indicator may be tested by instructions to determine when the complete record is ready for processing.

A writing operation works in the same way in reverse. Information is transmitted from memory, five characters at a time, and placed in the data synchronizer input buffer. From there it is transferred to the output buffer and to the tape, character by character.

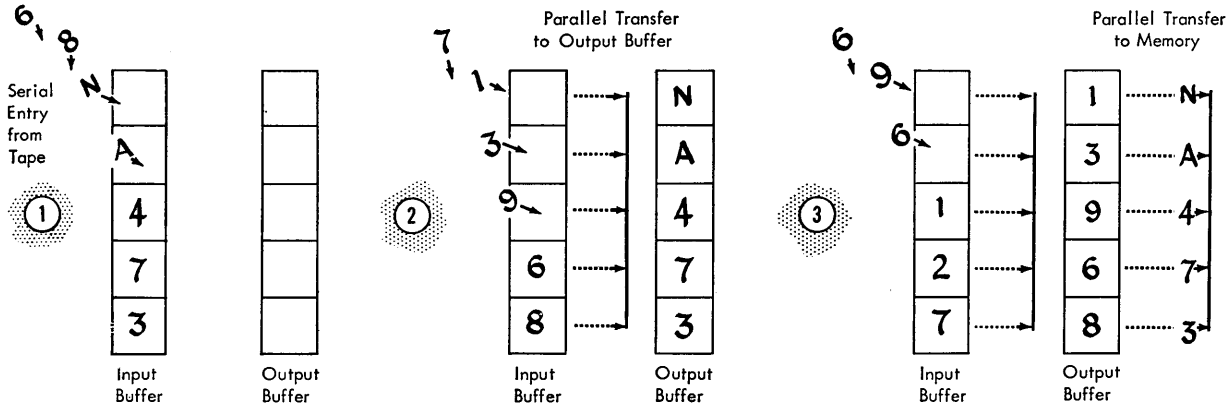


Figure 54. Data Synchronizer, Input and Output Buffers

During writing, as in reading, the processing of other data may continue in the CPU, interrupted periodically for nine microseconds as each group of five characters is transmitted from memory to the input buffer.

Data synchronizers offer the following advantages:

1. The efficiency of the 705 system is greatly increased by overlapping the functions of reading or writing and processing. Still greater efficiency is gained if two or more DS's are used.

2. The time that the CPU is occupied with the receiving and disposing of information, by way of tapes, is greatly reduced because data are transferred between the DS and memory at the rate of nine microseconds per group of five characters. This transfer time is so short that it does not materially affect total job time. For example, one minute is added to a complete procedure for transfer of 33,000 records of 1,000 characters each.

3. Writing need not be delayed by reading, nor reading delayed by writing, since these operations can occur independently in separate data synchronizers.

4. No practical limit is placed upon the length of records that may be read from, or written onto, tape.

5. A tape can be reading or writing through a DS while the system is:

   a. Processing in the CPU or reading from cards.
   b. Punching cards or printing.
   c. Reading or writing another tape through another DS.

Tape Record Checking; Writing

All information written on magnetic tape is automatically checked for validity. The method of checking differs somewhat between the two models of tape units – the 727 or the 729.

When the 727 is used in the system, the information in memory is transmitted through the IBM 754 Tape Control to the write head of the previously selected tape unit. The records are then written magnetically, character by character, in 705 code in the seven tape tracks.

As each character is sent from memory, it is checked in a character register in the CPU to make sure that the total number of 1's representing that character is always even (Figure 56). If it is not even, a machine check indicator is turned on to signal the error. However, the complete record is written without interruption.

The impulses to the write head from memory are then returned as "echo" impulses to a register in the tape control, in the same pattern in which they were received. These impulses, representing the character just written, are also checked for an even number of bits or 1's. After the writing operation has been completed, the condition of this register is checked and, if an error has been sensed, the read-write check indicator is turned on to signal that an incorrect record is on the tape.

Both machine and read-write check indicators may now be tested by instructions. The action to be taken is entirely governed by further instructions and may vary, depending upon the requirements of the procedure. If the error occurred after the record left memory, the machine may be instructed to backspace the tape and rewrite, again checking to determine if the operation was properly performed on the second try. If it is, normal operation can then continue automatically. The number of rewrites can be predetermined – for example, the machine can be instructed to attempt correction up to three times. If the error still persists, operation halts and manual intervention by the operator is necessary. In effect, the machine automatically senses its errors, attempts to make a correction and, if successful, continues operation without stopping.

If an error has developed in memory, however, any rewriting operation cannot correct the condition. In this case, the machine is normally instructed to stop so that some manual correction to the record can be made. When this is not feasible, the error record may be printed for later correction in subsequent processing.

During writing on tape, an odd-even indication of the total number of 1's in each separate track is stored in the tape control. At the end of every block of data, an extra 1 is written where necessary to make the total count in each track even. This forms a check character at the end of the record. The check charac-

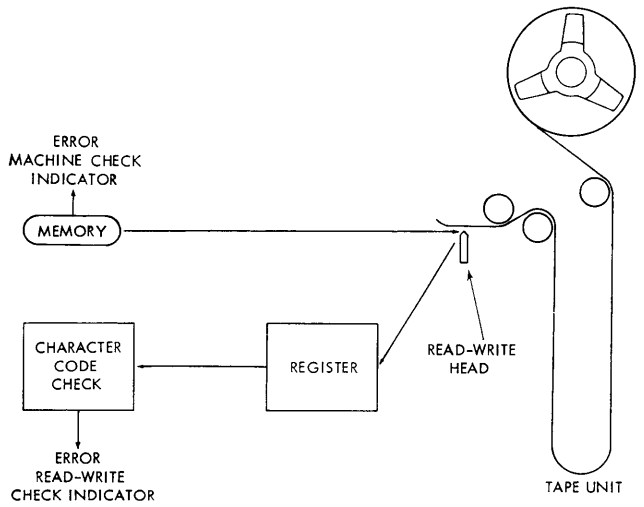


Figure 56. Tape Output Check

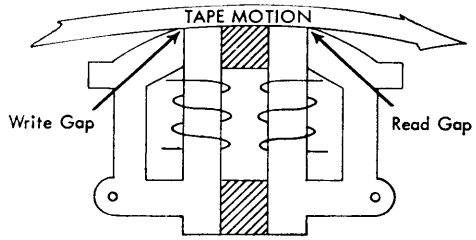


Figure 57. Two-Gap Read-Write Head

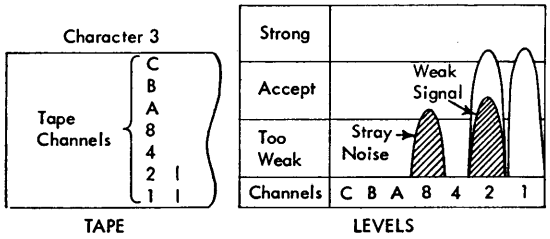


Figure 58. Checking Bit Signals while Writing

ter is used for further checking whenever the record is read.

Checking methods on the 729 I and III have been improved with the use of a "two gap" read-write head (Figure 57). In these units, one head is used for writing, the other for reading.

During the writing operation, all information recorded on the tape is immediately read back for an automatic validity check on each character. In addition, a critical analysis is made of the signal strength of the written record to assure future reading reliability (Figure 58). This analysis determines that:

1. Signal strength is well above the given tolerances required to read the tape record later.
2. No unwanted signal or other interference is present in the record area.

When an error is detected, the machine can institute automatic corrective action under control of instructions. This action is the same as that using the 727 units, except that it can also include skipping a defective section of the tape. The error record is erased and the section skipped is five or six inches in length. In this manner, the writing of records on any defective portions of a tape is prevented without any necessity for discarding the entire tape. When this same portion of the tape is read, the section is again automatically skipped by the 729.

Tape Record Checking; Reading

The 727 tape unit automatically checks in two ways all information read from tape (Figure 59).

A character code check is made on each character as it enters memory from the tape unit. This is a vertical check to insure that an even number of 1's will be stored for each character transmitted from the read head. An error turns on the read-write check indicator.

A tape record consists of a sequence of characters along the tape followed by a check character. Each time a character is read, it is set into seven binary triggers in the tape control. These seven triggers keep an odd-even count, by bit, of the entire record. At the end of the record, the check character is also read into the same register. Since the check character (as formed during the write operation) should contain a 1 wherever necessary to make the total number of bits in each track even, the sum of bits in the record plus the check character should always be even. If it is not even, a single bit or an odd number of bits were

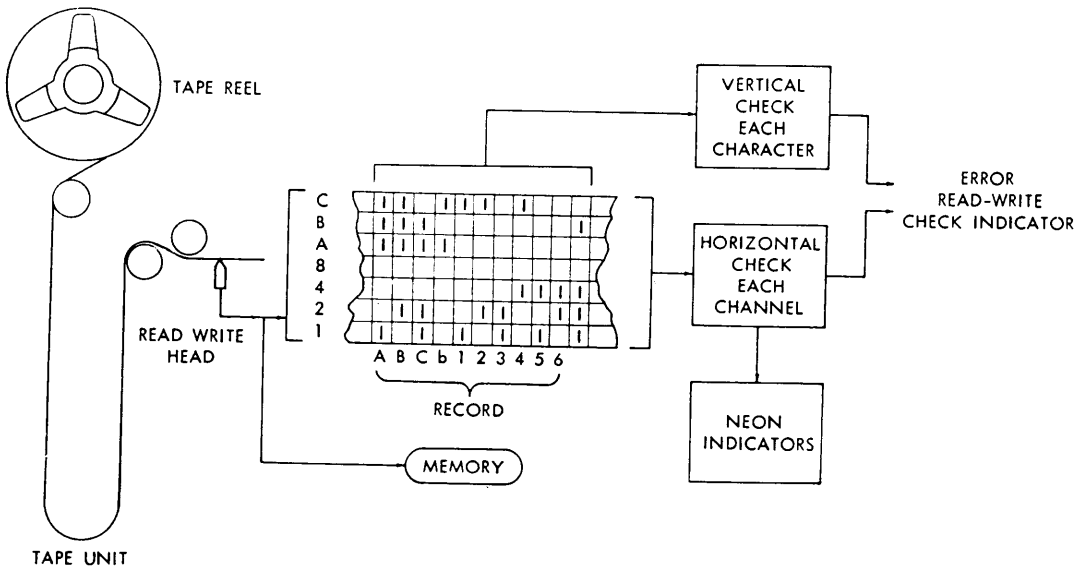


Figure 59. Tape Input Check

lost or picked up in reading that record. An error turns on the read-write check indicator. The check character is not transmitted to memory. Again, corrective action can be automatically taken by the machine under the control of instructions. The tape can be backspaced and reread. If the error is corrected, operation continues automatically. If the error persists, operation can be halted for manual intervention or repairs.

During a 729 read operation, each character is placed in two registers in the data synchronizer, a high and a low register (Figure 60). First, the low register is checked. If all the signals are acceptable, i.e., if the character is valid, it is sent to memory without regard for the contents of the high register. If the low register shows an error, however, the contents of the high register are superimposed upon the low register in an attempt to correct the error. The effect is the same as if signals missing from the low register were supplied from the high register. The low register is then rechecked and, if correct, its contents are sent to memory. If it is invalid, a PCT data check indicator is turned on which can then be tested by instruction. Corrective action is essentially as explained for the 727 tape unit.

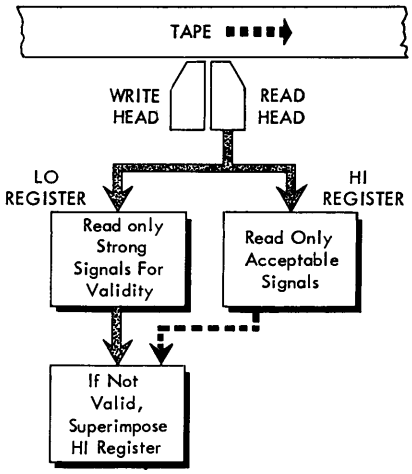


Figure 60. Read Checking, IBM 729 III

Figure 61 is a comparison of tape unit statistics.

		727	729 I	729 II	729 III	729 IV
Tape speed in inches per second		75	75	75	112.5	112.5
Recording density per inch		200	200	200	556	556
Character rate per second		15000	15000	15000	62500	62500
Record gap in inches		3/4	3/4	3/4	3/4	3/4
Record gap time in milliseconds		10.8	10.8	10.8	7.3	7.3
Maximum number of tape units per tape control device.	IBM 752	10				
	IBM 753	10				
	IBM 754	10				
	IBM 755		8			
	IBM 760	2				
	IBM 766		8			
	IBM 767 (1)		5		5	
	IBM 774	1 or	1			
	IBM 777	8				
	IBM 7607 (2)			10	or	10
Model of tape unit used with each IBM system.	Aux. card-to-tape (3)	x	x			
	Aux. tape-to-card	x	x			
	Aux. tape-to-printer					
	IBM 717	x	x			
	IBM 720A-730A	x	x			
	IBM 650	x				
	IBM 702	x				
	IBM 704	x				
	IBM 705 I or II	x				
	IBM 705 III (1)		x		x	
	IBM 709		x			
	IBM 7070			x		x
	IBM 7090			x		x

(1) A maximum of ten 729 I tape units may be used if no 729 III tape units are used; otherwise a maximum of five of each type must be used.
(2) Ten intermixed tape units may be used on each IBM 7607.
(3) Dual level sensing is not active.

Figure 61. Comparison of Tape Unit Characteristics

Card Reader

IBM cards are important source documents in the data processing system. The 705 can further process punched cards prepared by other IBM equipment and can record the results on magnetic tape, drum, other punched cards, or printed records. The IBM 714 Card Reader handles input data at the rate of 250 cards per minute (Figure 62). It reads standard 80-column cards.

Figure 63 is a schematic showing the flow of information from the card stations to memory. The first station after the card feed is used for checking and control. At the second, or reading station, punched information is transmitted to a 92-position record storage unit where it is held until called for by a read instruction from the 705.

Record storage is an intermediate or buffer storage between the read station and memory.

When the card reader is selected, a reading instruction transmits the contents of record storage to the specified address in memory. This operation clears record storage for receiving the next card. After transmission, further instructions can be executed while the following card is being read into record storage. For example, calculations may be made from one card and the results sent to other output units. These units may, in turn, begin writing while record storage is being filled from the next card. In this way, the 705 continues with other work without delay during the comparatively slow mechanical operation of card feeding.

A control panel (Figure 64) provides additional flexibility in the card reader. The function of the panel is similar to that of panels in other IBM card

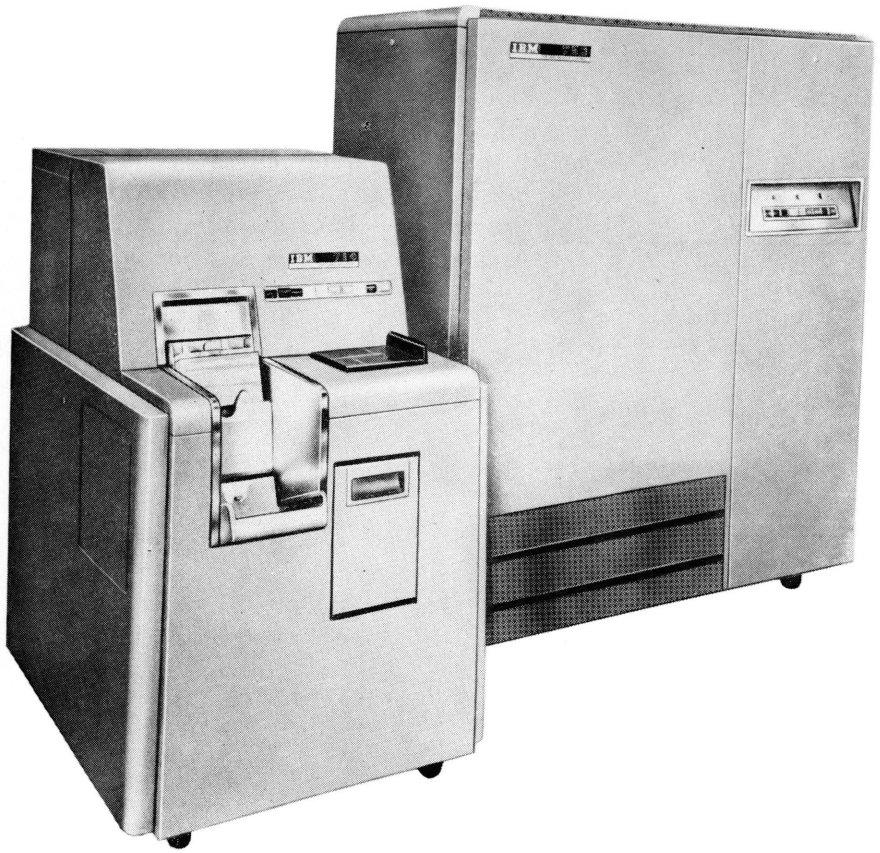


Figure 62. IBM 714 Card Reader and IBM 759 Card Reader Control

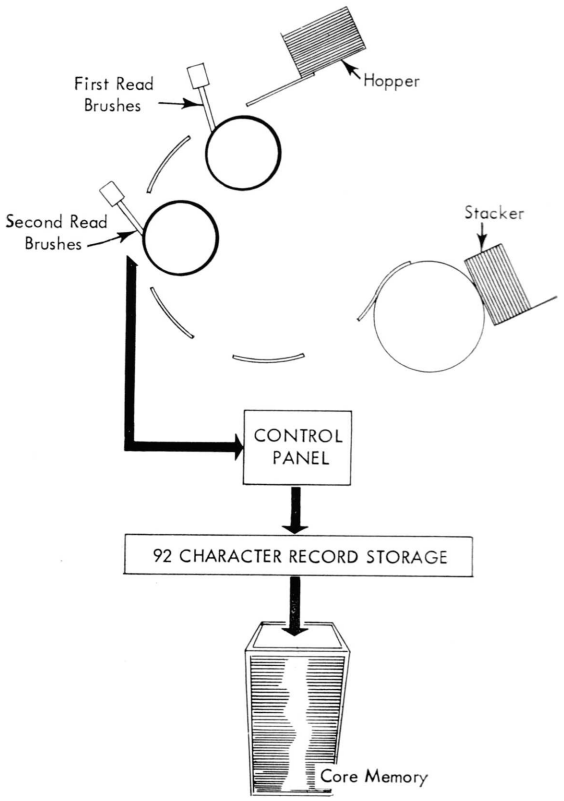


Figure 63. Data Flow from Cards to Memory

equipment. It may be used in one or more of the following ways:

1. Selection of only those fields to be actually used in processing (Figure 65).
2. Rearrangement of fields in any desired order. The record can be made to conform with the arrangement of other card or tape records without repunching cards (Figure 66).
3. Signing numerical fields with indication of plus or minus. Normally card fields are punched with a hole in the 11 row to indicate minus; no punching indicates plus. Since all arithmetic fields in memory must be signed for either plus or minus, the signs may be emitted from the panel without special card processing (Figure 67).

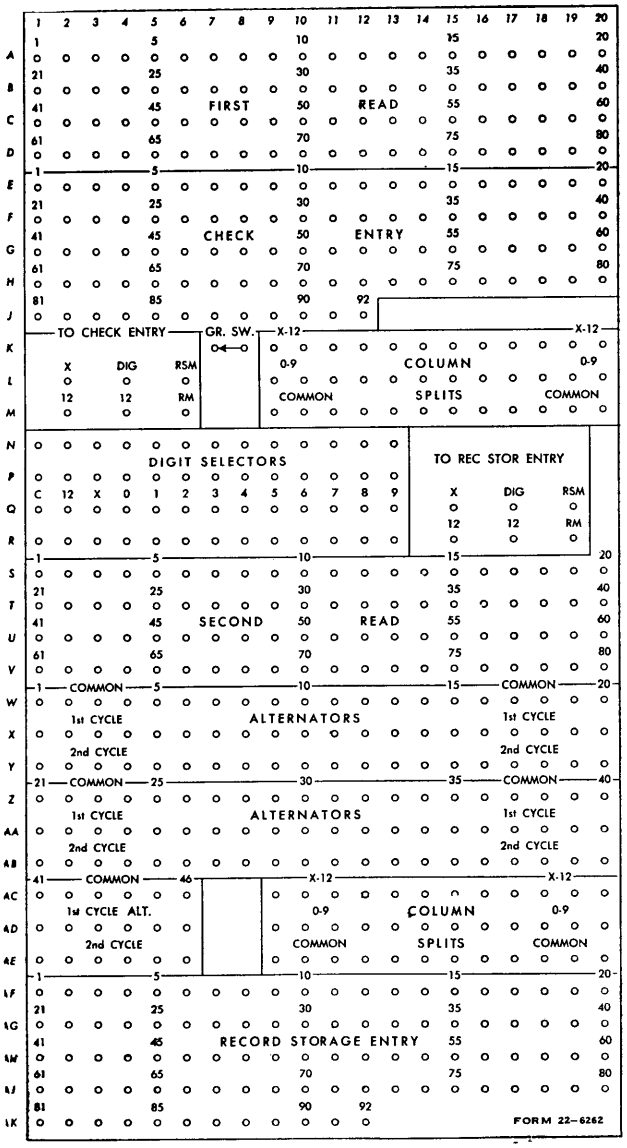


Figure 64. Card Reader Control Panel

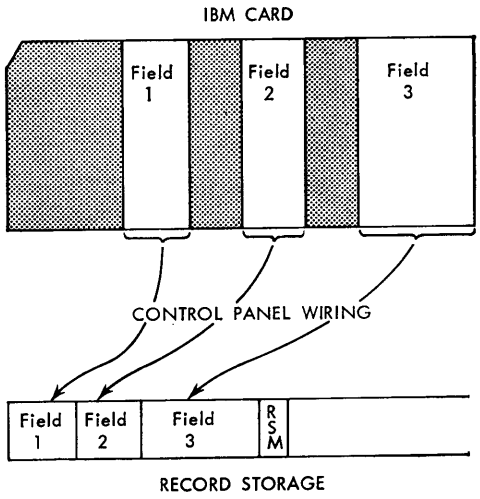


Figure 65. Field Selection, Card Reader

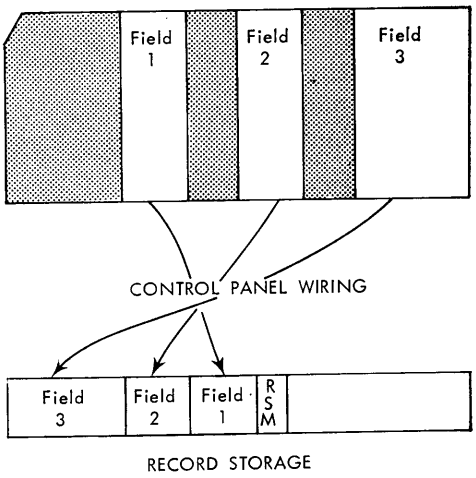


Figure 66. Field Rearrangement, Card Reader

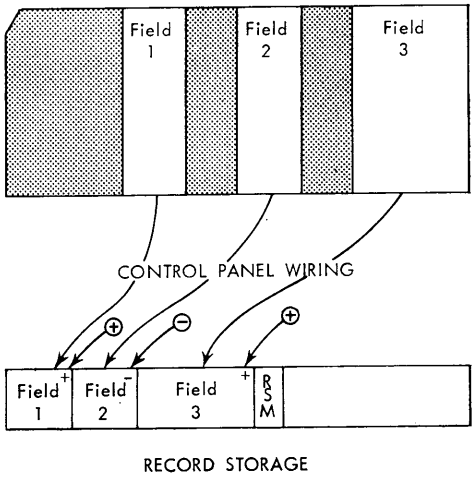


Figure 67. Signing Fields, Card Reader

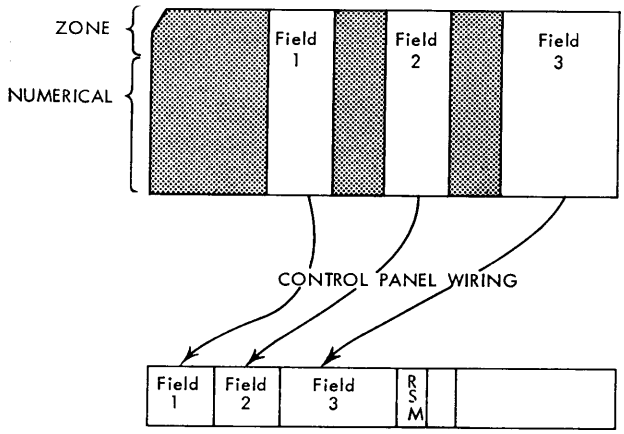


Figure 68. Zone Elimination, Card Reader

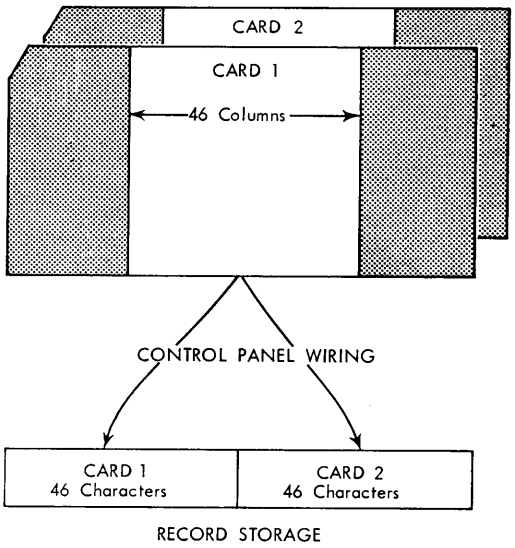


Figure 70. Grouping Records, Card Reader

4. Elimination of unwanted zone punching (Figure 68).
5. Emitting of constants, such as date, factors for calculation, special characters, and so on (Figure 69).

A record storage mark is normally wired on the control panel into the position of record storage immediately following the last significant character of the card record. The mark then limits the length of the record read into memory. For example, if only 19 columns are selected from the card, the mark can be wired into the 20th position of record storage. Only 19 characters are then read into memory. If the mark is not wired, a 92-position record always enters memory. Unused positions of record storage then appear as blanks. The storage mark is not transmitted to memory.

The group mark character may also be emitted from the control panel. This character is used to define the limit of the record in memory when writing.

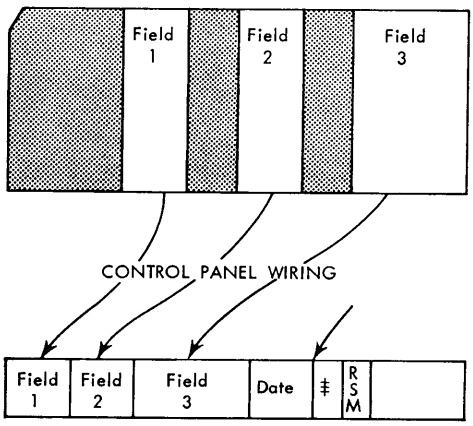


Figure 69. Emitting Constants, Card Reader

A grouping feature permits the reading of two cards into record storage for transmission to memory as a multiple record. As many as 46 characters may be read from each card. However, the same card columns must be used for all cards (Figure 70).

Each card reader attached to the 705 requires an IBM 759 Card Reader Control. Like all other components of the system, the card readers are assigned addresses, 0100-0199. The first two digits, 01, always specify card reading; the last two digits, the particular reader to be used.

Card Reader End-of-File Operations

When the card hopper empties while reading records into the 705, the feed stops immediately. More cards may then be placed in the hopper and, if the start key is depressed, operation continues normally.

However, if there are no more cards in the file and if the start key is depressed, an end-of-file sequence is set up. Feeding continues under control of the CPU until the last record has been transmitted from record storage. Another feed cycle then occurs, during which nothing reads into the storage unit. The next time CPU sends a select and read call, the input-output indicator is turned on, signaling that the card reader is at the end of file.

The machine can then be instructed to stop, to continue operations using other input-output devices, or to execute any special instructions pertaining to the end of the job.

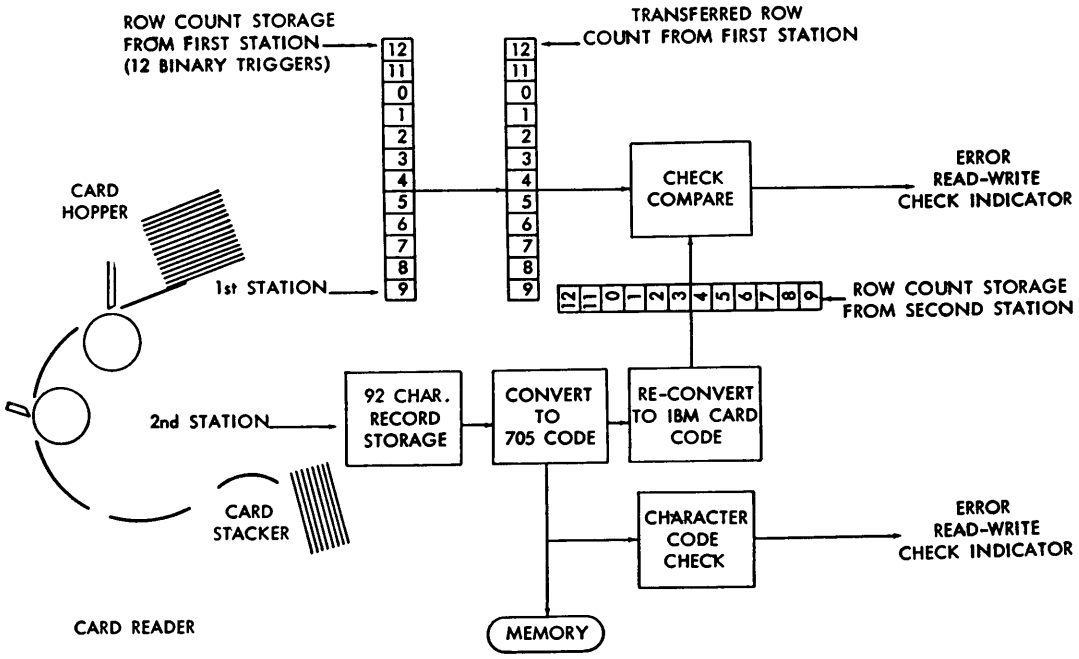


Figure 71. Card Input Check

Card Reader Checks

The 705 checks automatically in two ways all information read from IBM cards (Figure 71).

Horizontal Check

At the first card station, the number of holes in each horizontal row of the card is determined to be odd or even, that is, the total number of 9's, the total number of 8's, 7's, 6's, and so on. Only those columns are counted that are wired for entry to record storage.

The odd-even indication is stored, one row at a time, in a temporary storage device consisting of 12 binary triggers. Each trigger can indicate one of two possible conditions, odd or even.

When the card has passed the first station, the odd-even count of all 12 rows is transferred to a second set of triggers where it is retained during the next card cycle. The first set of triggers is then free to indicate the row count for the following card. The reading at the first station is used only for checking.

At the second station, the card columns wired from the control panel are stored in record storage. When a read instruction is given, the card record in record storage is converted to the 705 character code and is sent to memory. The 705 code is also reconverted to the IBM card code and an odd-even count is made of each row. This count is stored in a third set of triggers where it is then compared with the count obtained and stored when the card was read at the first station. A difference in the comparison turns on the read-write check indicator.

Character Check

The card record read from record storage into memory is also given a check for an even number of ones in each character. An error here also turns on the same read-write check indicator.

If either of the two error conditions just described occurs, the check indicator is turned on *after* the incorrect record is stored in memory. Instructions test the indicator and stop the machine if it is on. Several correction procedures are possible. The error card may be reread by taking it from the hopper and feeding it under manual control to record storage. The read instruction is then repeated and, if a correct record is transmitted, the machine continues normal operation under control of the program. In some cases, the record may be manually corrected in memory, or it may be printed as an error and omitted from processing depending upon the requirements of the procedure being operated upon.

Card Punch

The 705 can also be equipped with one or more card punches. Standard 80-column IBM cards are prepared as output records which can then be further processed by card-operated equipment. The IBM 722 Card Punch punches at the rate of 100 cards per minute. Each punch requires an IBM 758 Card Punch Control (Figure 72).

Figure 72. IBM 722 Card Punch and IBM 758 Card Punch Control

The record from memory is first transmitted to an 80-position magnetic core record storage. The record is then mechanically punched while the CPU continues with other instructions in the program. If the record in memory is longer than 80 characters, other operations of the CPU are delayed until two or more cards are punched for the complete record. Figure 73 is a schematic of the punch feed and the card record storage.

A record is sent to punch record storage in exactly the same arrangement in which it is to be punched; that is, the record is first set up in memory to fit established card fields. There is no control panel on the punch.

Addresses 0300-0399 are set aside for card punches; the digits 03 always signify a punching operation and the last two digits specify the particular unit to be used.

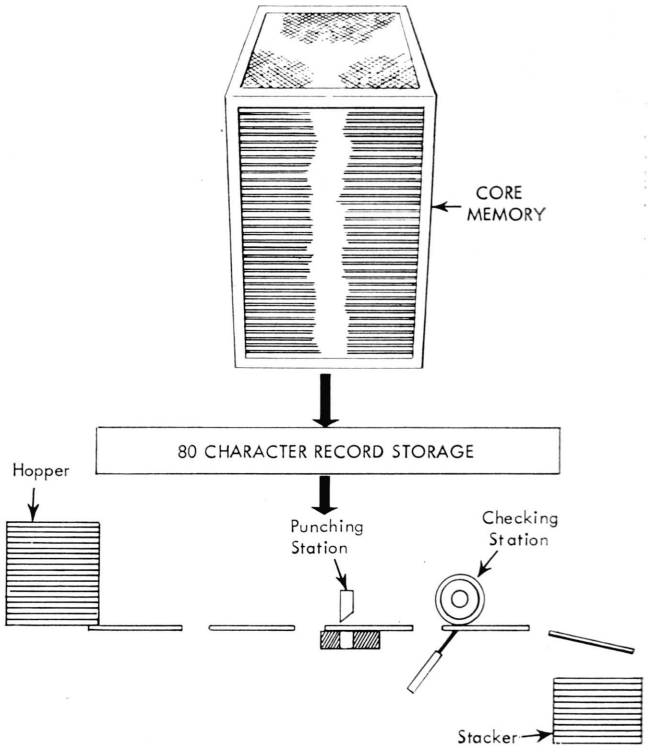


Figure 73. Data Flow from Memory to Cards

Card Punch Checking

All punching of IBM cards by the 705 is automatically checked in two ways (Figure 74).

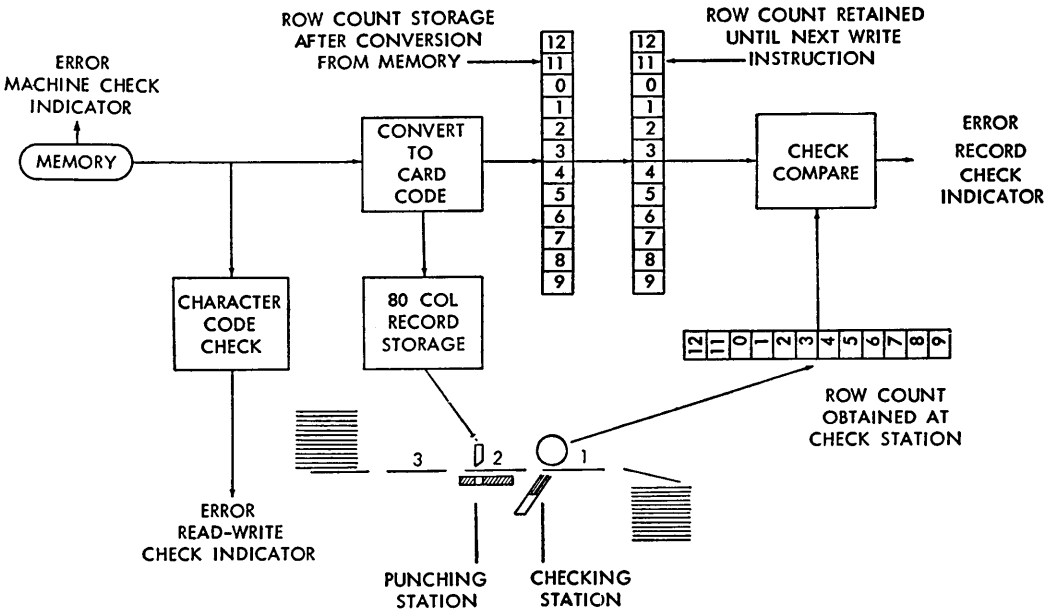


Figure 74. Card Output Check

Character Check

The record to be punched is given a character-by-character check when sent to record storage in the same way as for all data handled within the machine. An error in transmission turns on the read-write check indicator. In this case, it is turned on before the card is punched. If the indicator is then immediately tested by instruction, punching can be prevented to avoid making an error card. A blank card is advanced by this instruction. Record storage can be reloaded with the same record from memory and another test for error made. Punching can thus be prevented until record storage is correctly loaded.

However, if an error has developed in memory, the machine check indicator is also turned on, showing that corrective action must be taken by the operator or that the record must be set up again in memory.

Horizontal Check

The record to be punched from memory is first converted from 705 character code to the IBM card code. The entire record is placed in the 80-position record storage. During this operation, the machine determines whether the number of holes in each horizontal row of the card record is odd or even. The row count information is temporarily stored by an arrangement of 12 binary triggers. Each trigger can indicate one of two conditions, odd or even. Trigger storage is identical with that described for the card reader.

The row count from the first trigger is then transferred to a second set of triggers where it is retained until the next write instruction is given. The first triggers are then free to accept a row count from the next record to be punched. The card is punched at the punch station.

When the next write instruction for the punch is executed, the card passes the punch brushes. Again, the machine determines whether the number of holes in each horizontal row is odd or even. This row count from the punched card is transferred, one row at a time, to a third set of 12 triggers. Here it is compared with the count obtained and stored when the record was sent from memory. A difference in comparison turns on a record check indicator during the *next write instruction* to this punch unit. The error card is now the top card in the stacker.

No corrective action is possible at this point. However, a message may be printed to indicate that an error has occurred, bringing the condition to the attention of the operator.

IBM 717 Printer

The IBM 717 Printer is one of two types of printers available to produce printed reports directly from the central processing unit. The 717 writes at the rate of 150 lines per minute with lines as long as 120 characters. Each printer requires an IBM 757 Printer Control as shown in Figure 75.

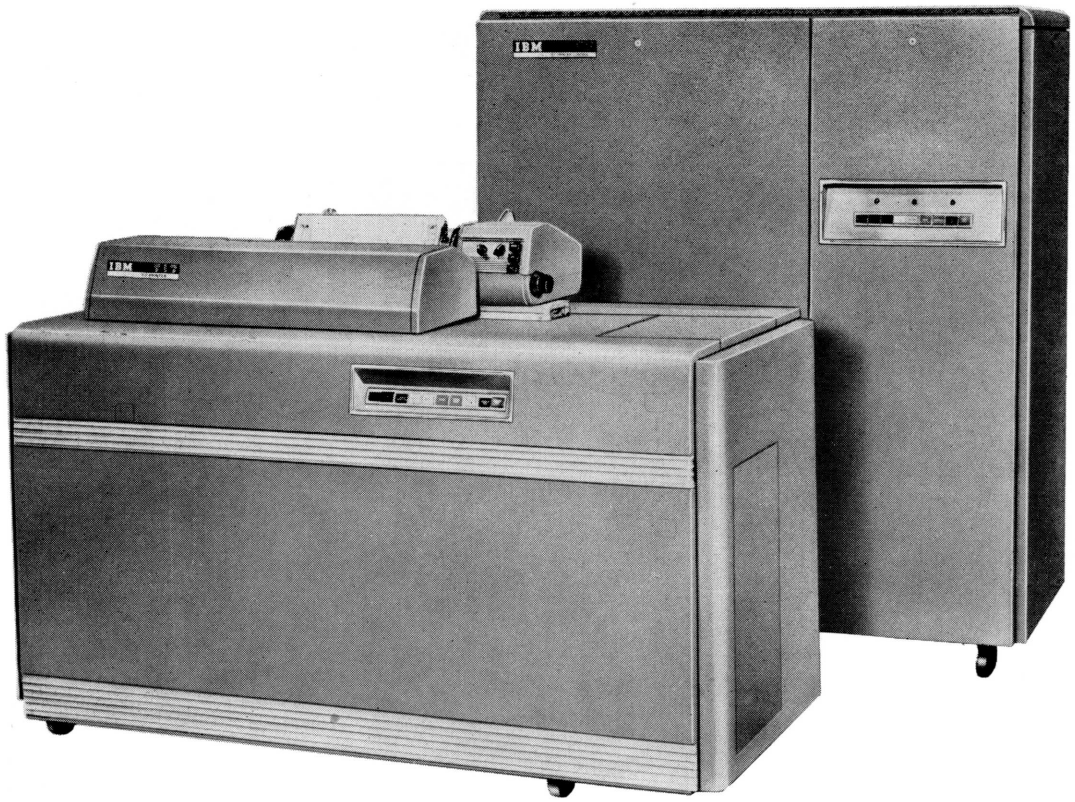


Figure 75. IBM 717 Printer and IBM 757 Printer Control

The printing unit of the 717 consists of 120 type wheels, one for each possible character position in the printed line. Each wheel is engraved around its face with 47 different characters (Figure 76), including all letters of the alphabet, numbers, and the special characters which can be handled by the 705 as punched or printed output. The mechanical operation of printing is the same as used in the IBM 407 Accounting Machine.

All records are printed from memory. When the printer is selected and instructed to write, the record is first transmitted to a 120-position printer record storage. From there, it is printed as one line of characters while the 705 continues with other instructions. Calculations, or reading and writing with other input-output devices, may take place while a line is being printed. If the record in memory is longer than 120 characters, other operations in the CPU are delayed until two or more lines are printed for the complete record.

There is no control panel on the 717. Therefore, all information to be printed must be arranged in memory to fit the report form. Instructions in the 705 can edit the record, performing the functions of deleting left zeros from numerical fields and placing of decimal points, dollar signs and the like. Totals are accumulated in the 705 and may be printed as separate records from memory.

Printer addresses are assigned from 0400 to 0499.

Automatic Carriage

An automatic carriage feeds and spaces forms for printing (Figure 77). The carriage is controlled by punched holes in a paper tape. The tape corresponds to the length of one or more forms. The punched holes start or stop the movement of the form at predetermined positions (Figure 78).

The carriage accommodates continuous forms to a maximum sheet length of 22 inches at six lines per inch or 16½ inches at eight lines per inch. The forms may be a maximum of 16¾ inches wide, including punched margins. Although forms of any size within these limits can be handled in the carriage, forms of standard sizes can be obtained more quickly and economically from the forms manufacturers.

Forms can also be designed to permit printing in practically any desired arrangement, and skipping to different sections of the form can be controlled by holes punched in the paper tape.

When one form is completely filled, it can be ejected and the next form can advance to the first printing

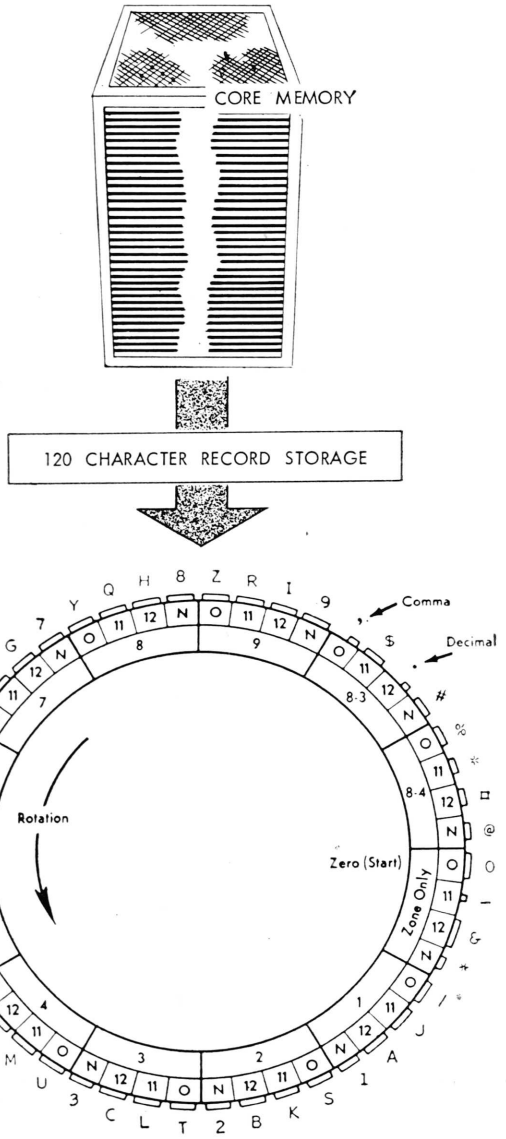


Figure 76. Schematic, Print Wheel

line or to the first body line. This "overflow skipping" is caused by sensing a punch in a specific position of the tape. If a group total occurs immediately after a record, the total may be programmed to print in some specific line on the form. The overflow punch in the tape can also be used to start other operations, if desired, before ejecting. For example, a total may be printed at the bottom of each page before advancing to the next form.

Two IBM forms tractors are available for the printer. The F2 is standard but the F4 may be specified in place of, or in addition to, the F2. Forms tractors are interchangeable. Each of the devices has two adjustable tractor-type pin feed units, one for each side of the form (Figure 79).

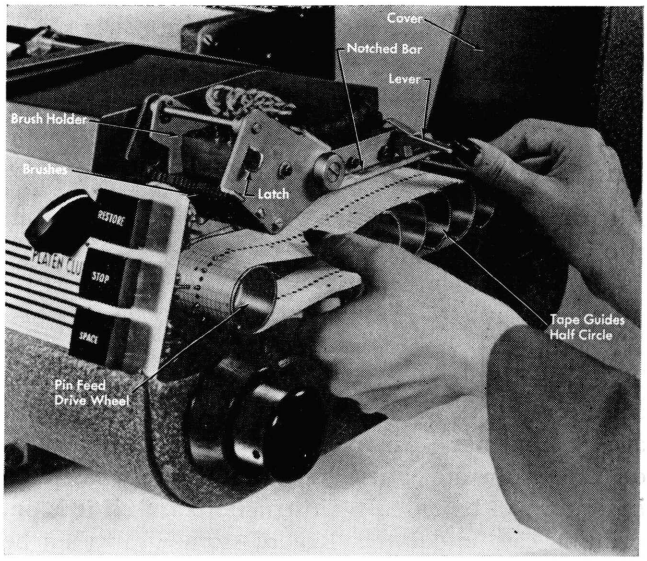


Figure 78. Tape in Carriage

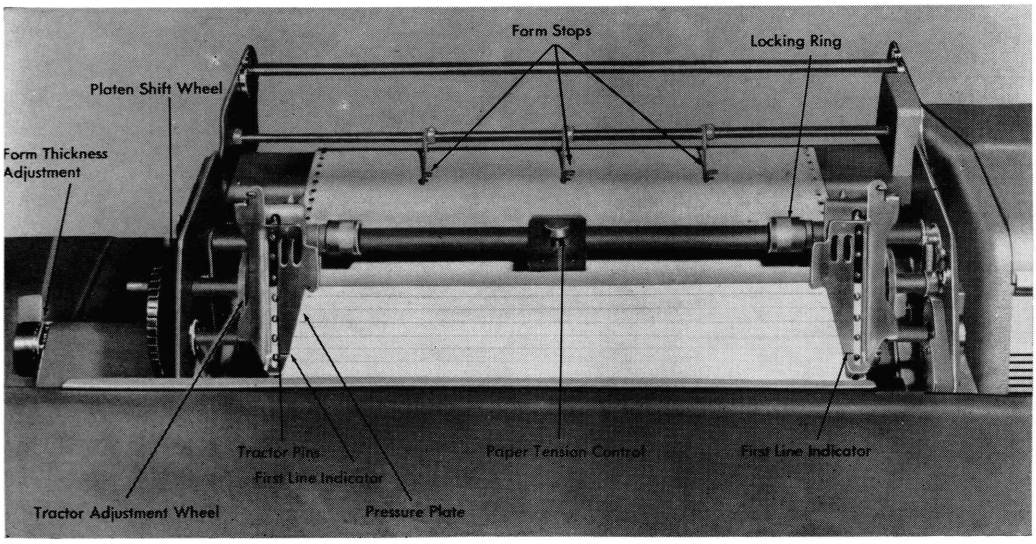


Figure 77. Form Feeding Carriage

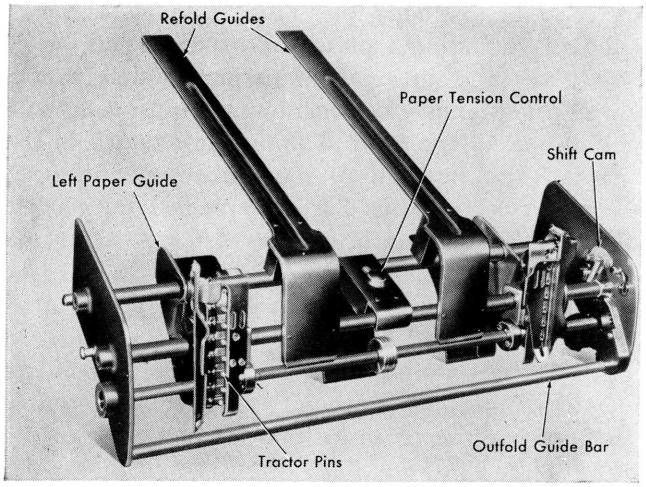


Figure 79. IBM Forms Tractor

The F2 provides a choice of spacing, either six or eight lines to the inch. The F4 provides a choice of either four or six lines to the inch.

Printer Checking for the IBM 717

All printing of data by the 717 printer is checked in two ways (Figure 80).

Character Check

The record to be printed is given a character-by-character code check when it is transmitted to record storage from memory. An error in this transmission turns on the read-write check indicator. In this case, the indicator can be tested by instruction, and, if it is on, printing can be delayed. Record storage can then be reloaded for a second try at printing. This corresponds to the punching delay when an error occurs between memory and the punch record storage. Printing can be prevented until record storage is correctly loaded – the number of tries can be limited by the program.

If a character code error exists in memory, the machine check indicator is also turned on. The record must then be corrected in memory by the operator or by rereading from the input unit, or the error can be noted for correction in later processing.

Horizontal Check

The record to be printed is converted from 705 character coding to the IBM card code and is stored in the 120-position printer record storage. At the same time that the converted record is entering storage, a horizontal row count of the digits 1-9 and the special character combinations 8-3 and 8-4 is stored in an odd-even register in the printer control. During printing, a horizontal row count of print wheel echo impulses is stored in another odd-even register also located in the printer control. The two registers are then compared. A difference turns on the record check indicator during the execution of the next write instruction involving that printer.

IBM 720A and 730A Printers

The IBM 720A and 730A Printers may also be attached to the 705 to produce high-speed printed output directly from information in memory (Figure 82). Each

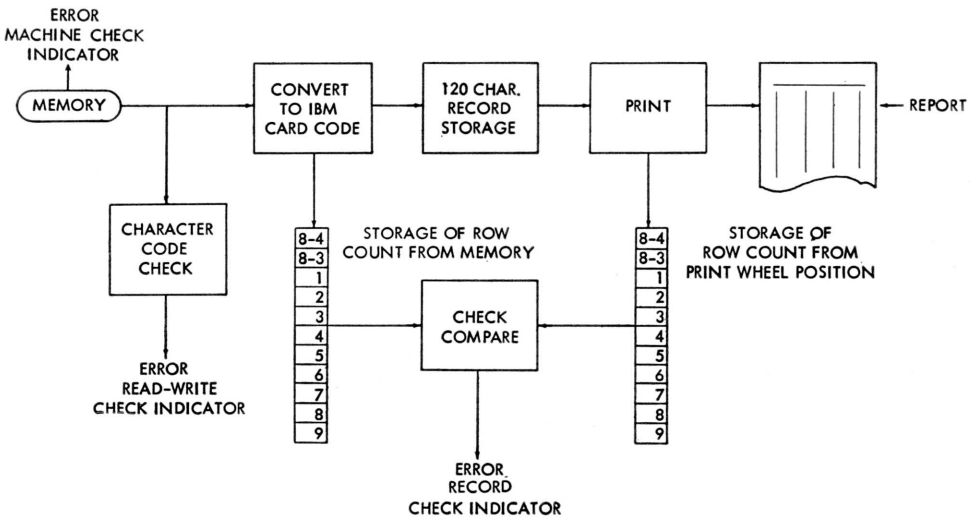


Figure 80. IBM 717 Printing Check

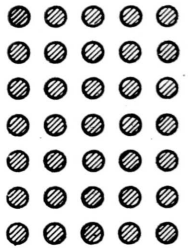


Figure 81

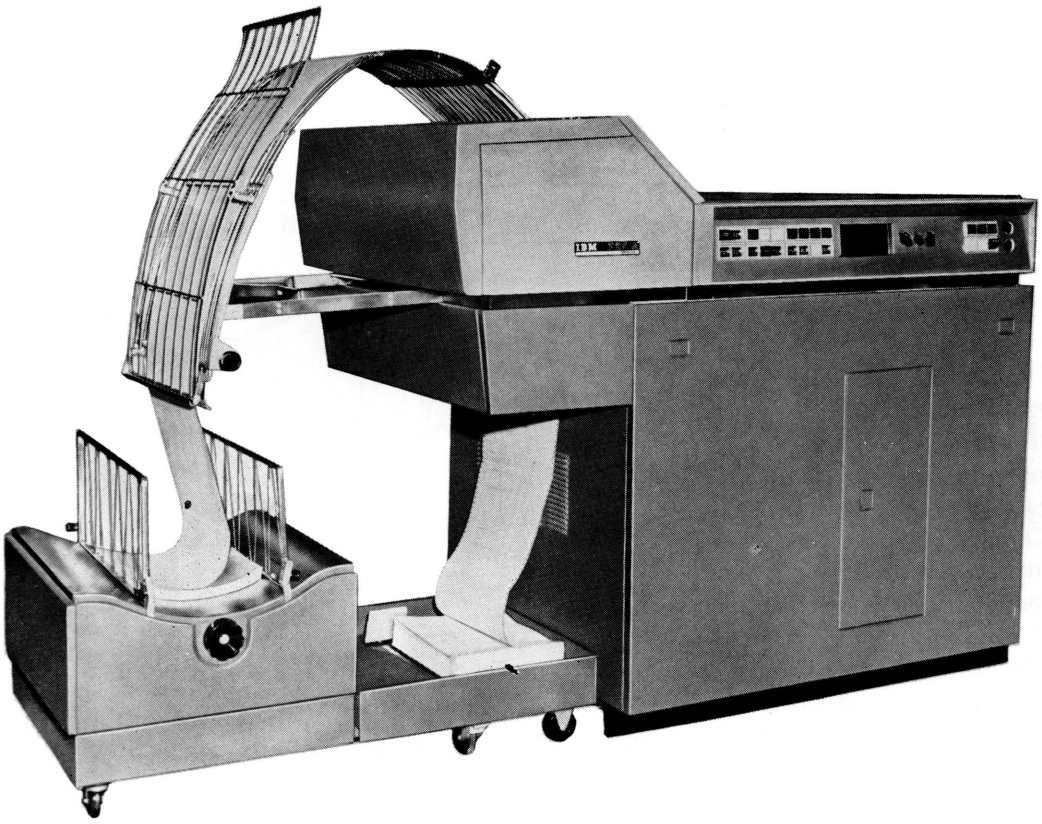


Figure 82. IBM 720A Printer

printer requires one 760 Control and Storage. The maximum speed of the 720A Printer is 500 lines per minute.

Maximum speed of the 730A is 1,000 lines per minute. Each line may contain as many as 120 characters.

The 720A or 730A uses no type wheels or bars. Instead of type faces that are already formed, each character is actually formed at the time of printing. The printing head is a matrix of 35 wires with each wire individually controlled. Characters are printed as a pattern of dots formed by the ends of the wires arranged in a five-by-seven rectangle (Figure 81). By extending selected wires, the patterns can take the shape of all the letters of the alphabet and the digits 0-9, as well as the special characters of punctuation and report printing. As each character is formed, the selected wires are pressed against an inked fabric ribbon to print on paper (Figure 83).

Form feeding is under control of an automatic hydraulically-operated carriage that provides quiet, precise movement of forms during skipping or spacing operations. Line spacing of either six or eight lines per inch is under control of a manual shift lever on the carriage. Single, double, or triple spacing while maintaining maximum printing speeds is under control of a punched paper tape that determines the point at

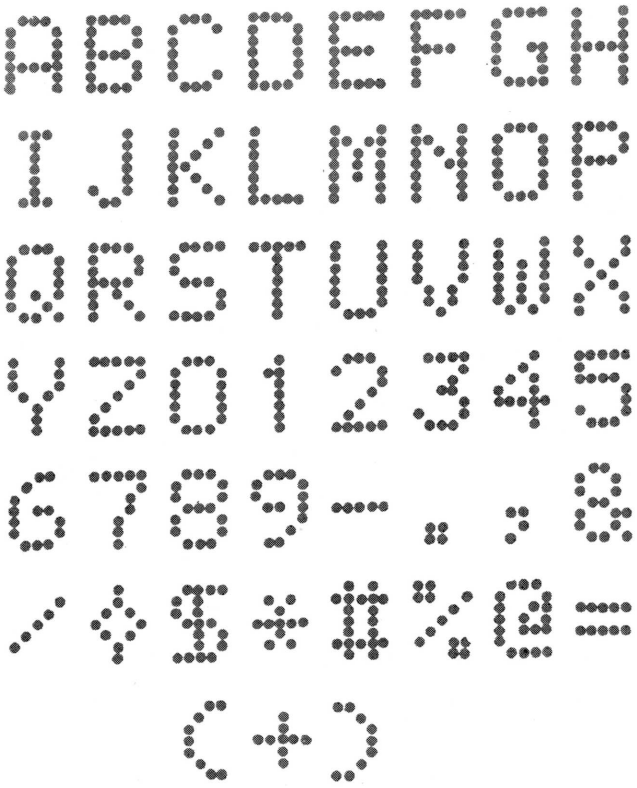


Figure 83. Wire Printing Dot Patterns

which skipping stops, or the point at which overflow begins. Forms from four to 20⅜ inches in width can be handled with a width of printing area up to 16¾ inches. The maximum form length is 22 inches for spacing of six lines to the inch, and 16½ inches for spacing of eight lines to the inch.

IBM 760 Control and Storage

The IBM 760 Control and Storage (Figure 84) provides intermediate record storage of up to 1,000 characters. Records can be stored either from the 705 memory or from tape during independent printer operation. Records are printed from storage in exactly the same order in which they are stored. Character arrangement within each line is, therefore, entirely controlled by arranging the output record in memory before printing.

The accuracy of record handling is checked first as the record enters storage from memory, and again as it is transferred to the printer. Records entering storage are given a vertical character check and a longitudinal record check. Records sent to the printer are given a character check for accuracy of print setup, a longitudinal record check, a test that a character is transmitted for every print position, and a test that the printer is in step with the output from the storage unit. Error indication is provided which may be tested by instructions in the 705 program.

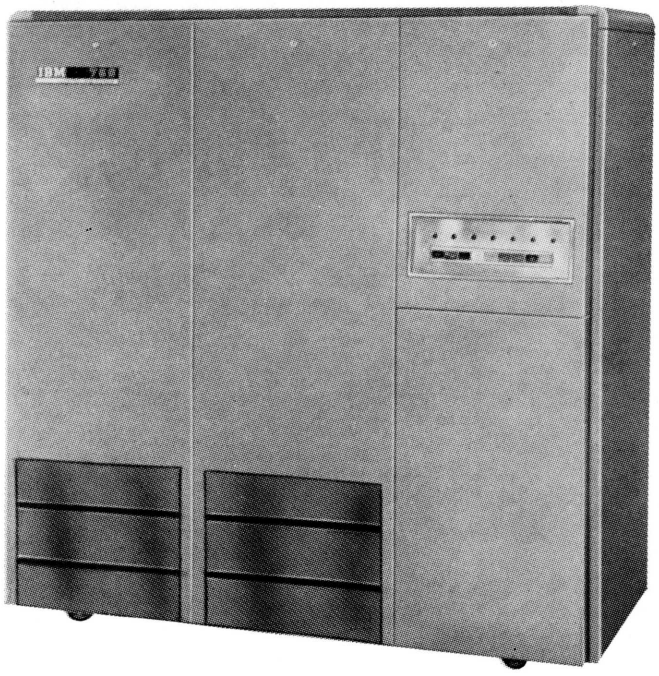


Figure 84. IBM 760 Control and Storage

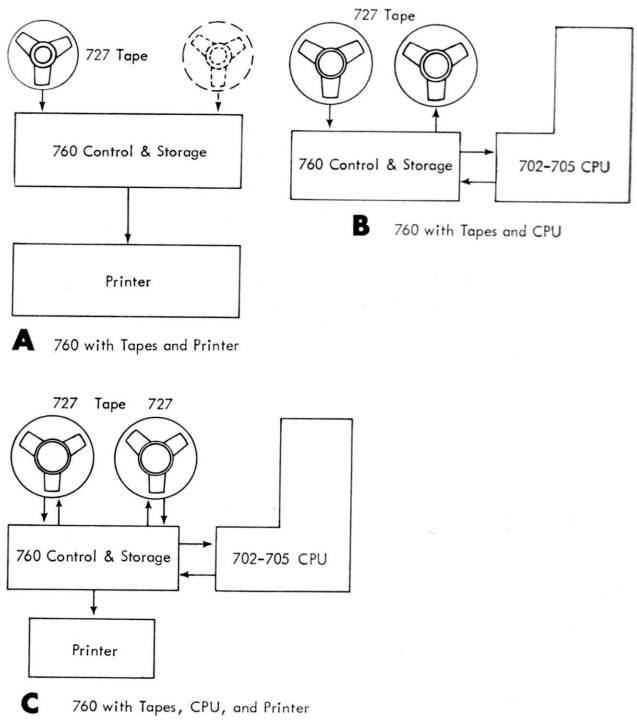


Figure 85. Schematic, IBM 760 and Associated Equipment

A check is also made that records do not exceed the capacity of storage. If an overflow is detected, the machine stops with an overflow indication light on the 760.

The IBM 760 Control and Storage is the center of operation in a number of arrangements of units, including the printers. Other arrangements are possible with tape units used as input or output through the intermediate storage of the 760 (Figures 85A, B, C). Tape-to-tape processing may also be accomplished without connecting the printer. Complete information concerning the printers and 760 is published in the *IBM 720A, 730A, 735, 760 General Information Manual,* Form 222-6768.

Console

The operator's console is a separate unit of the 705 system. It may be placed at any convenient position in the installation within the restrictions of the input-output cable lengths. Figure 86 shows a 705 system as it might be arranged so that all units are readily supervised from the console.

Four main types of controls are provided (Figures 87, 88).

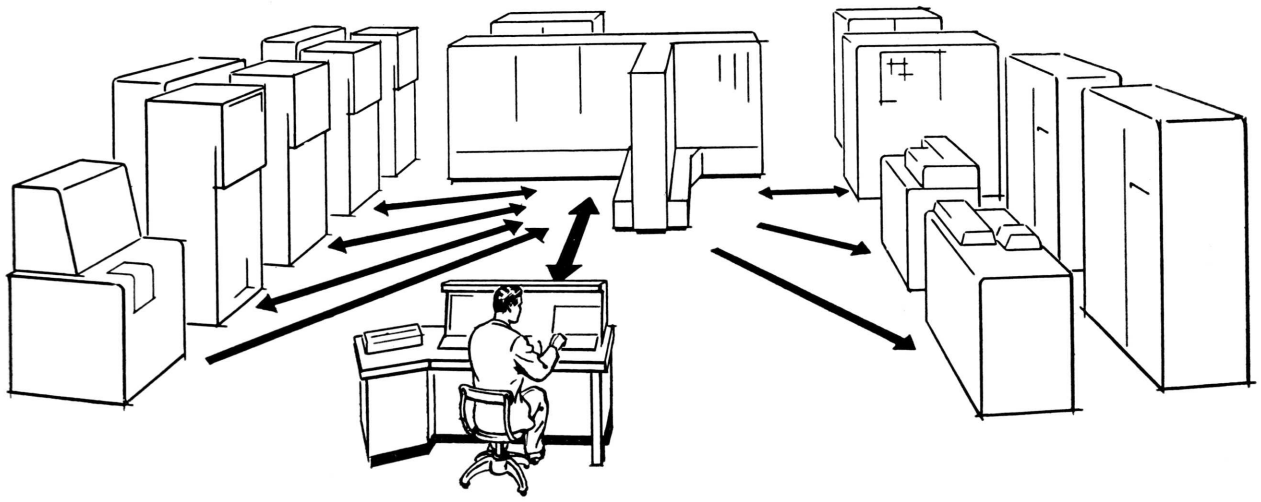


Figure 86. Schematic, Operator's Console

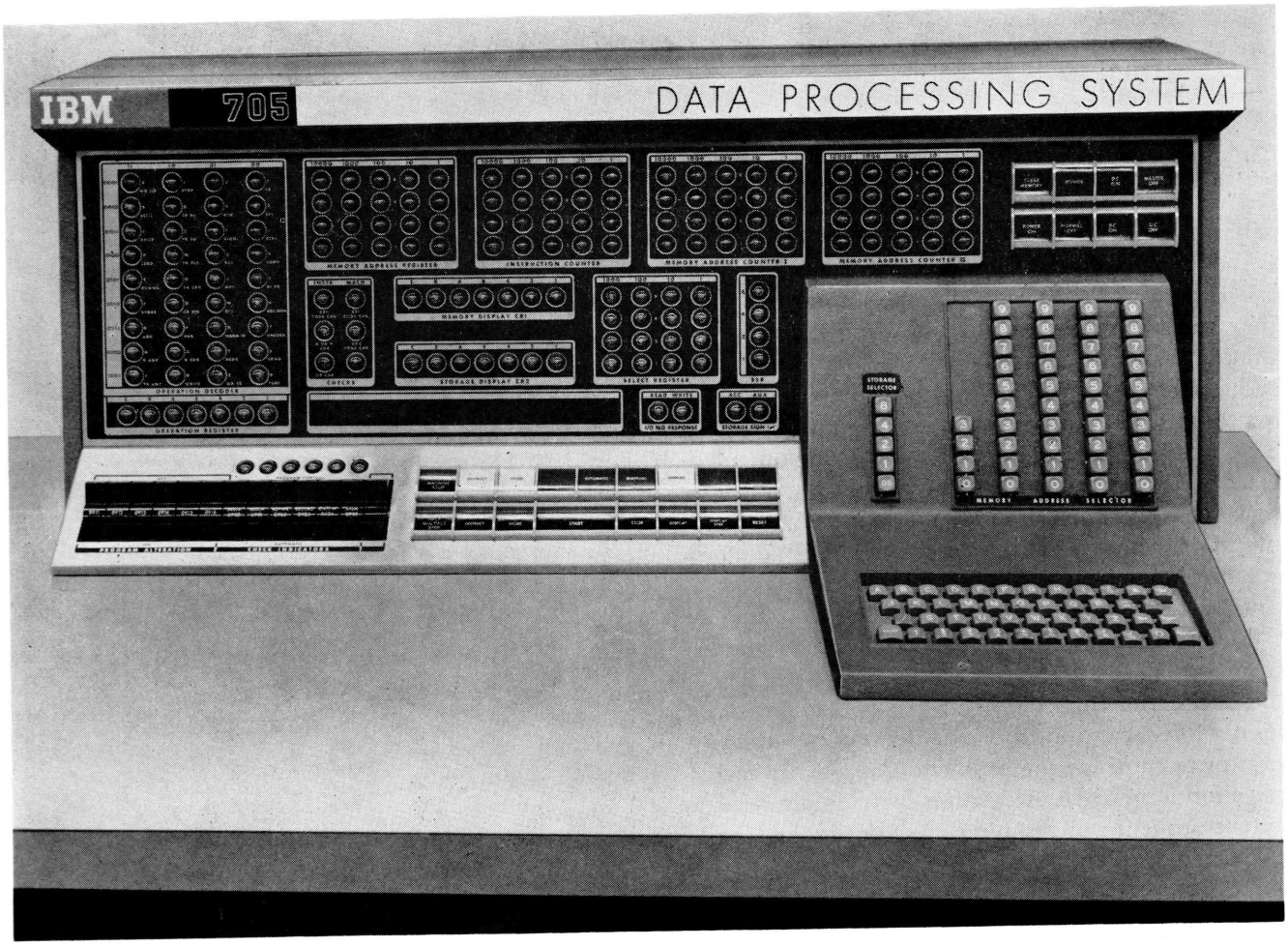


Figure 87. IBM 705 I and II Operator's Console

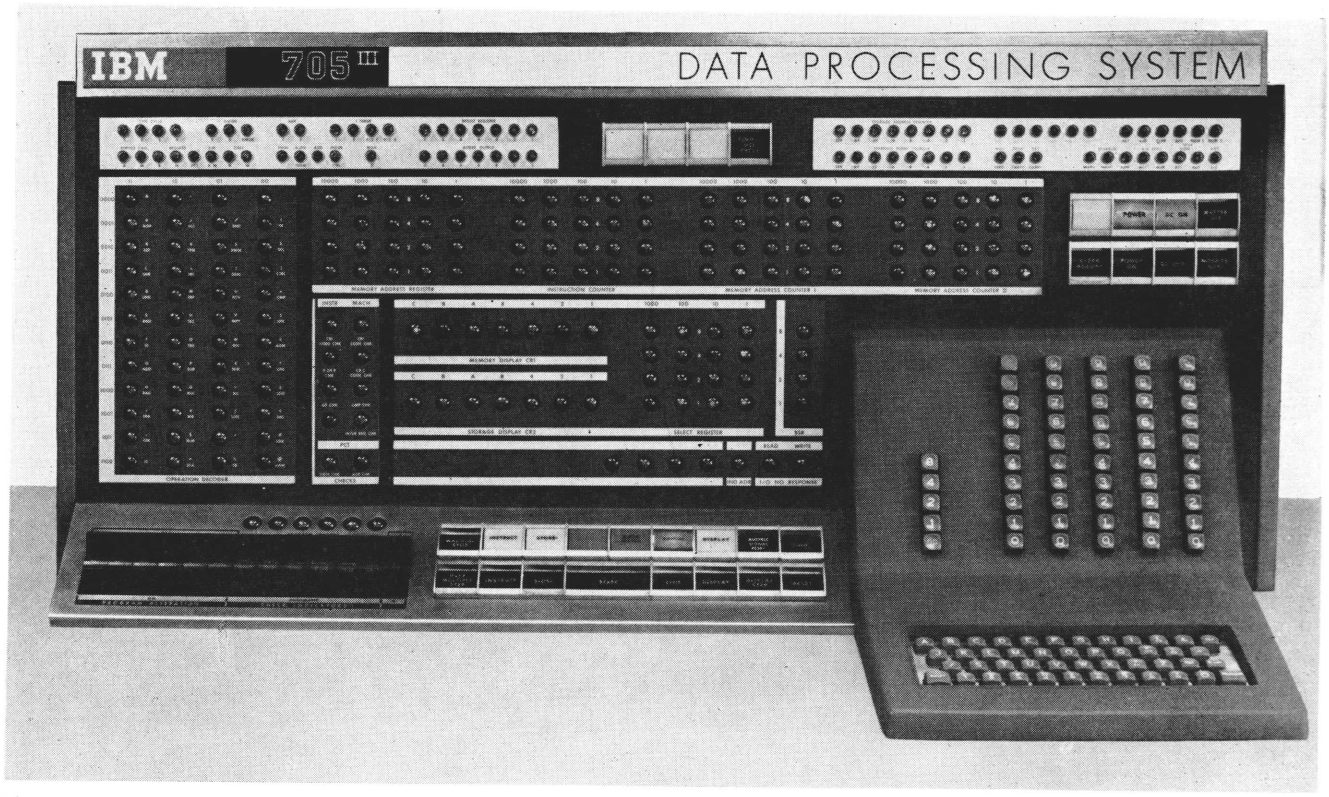


Figure 88. IBM 705 III Operator's Console

1. Neon lights and indicators enable the operator to display the contents of memory, character by character, or to display the contents of accumulator or auxiliary storage. Under manual control, a program may be stepped, instruction by instruction, in slow motion as an aid to tracing the operation of the machine. Neons show the instruction being executed, the operation being performed, and the status of other registers and counters. Neons also display the type of error condition that may occur.

2. Operating keys turn the power on or off; place the machine in automatic, manual, display, or store status; reset checking circuits and counters; clear the contents of memory; and clear accumulator and auxiliary storage. Information may be stored, character by character, directly into memory or instructions may be set up on the console and executed under manual control.

3. Checking switches provide the operator with an option either to stop the machine when an error occurs or to continue the problem if automatic corrective action is possible. Alteration switches are used to vary the sequence of machine operation according to predetermined plan.

4. A keyboard is used for either manual entry of data into memory or for keying the instructions to be executed under manual control (Figure 89).

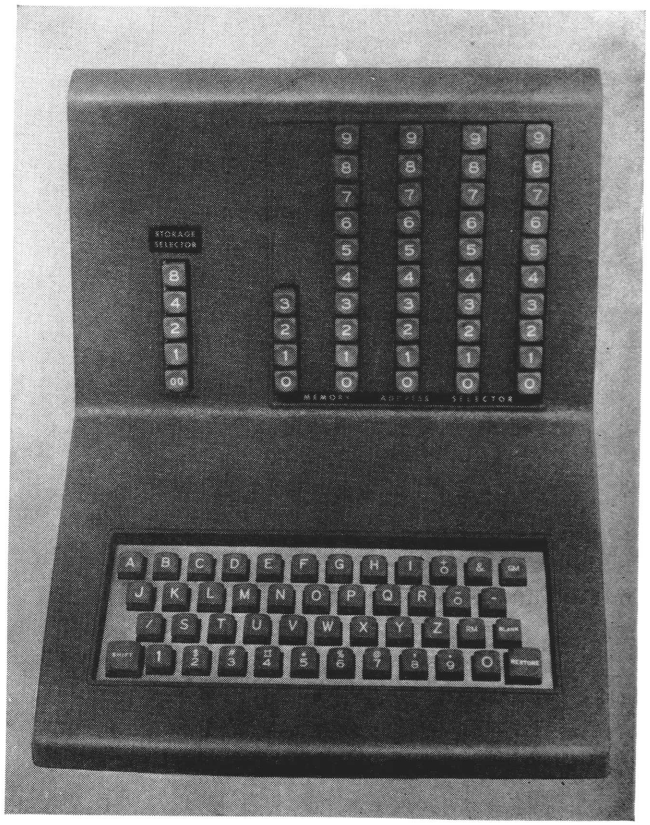


Figure 89. IBM 705 Keyboard

On the 705 III console, an audible signal sounds when the machine halts, calling the operator's attention to the fact that the system is idle. Loading of the program has also been simplified on the 705 III.

Typewriter

The 705 is equipped with a typewriter located to the left of the operator's console (Figure 90). The typewriter also operates under program control and prints directly from memory, one character at a time, at about 600 characters per minute.

The typewriter is used mainly for communication between the machine and the operator. Special conditions encountered during the course of the program, including end-of-file, exception records, and error records, may be noted on the typewriter. Accounting control totals, batch totals, or other control information can be supplied at predetermined intervals during a problem, to aid in over-all control of an application.

Other messages, prestored in memory, may be typed to indicate type or location of error, end of job, or to log the progress of the machine while running an application. Portions of memory may also be typed out as desired by manual instruction from the keyboard.

The 705 is normally equipped with only one typewriter. The assigned address is 0500.

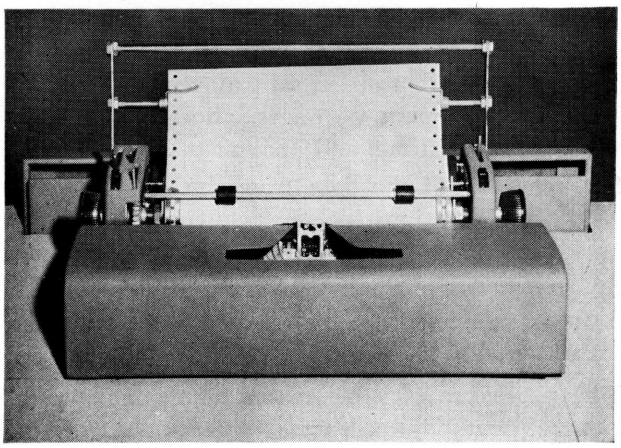


Figure 90. Typewriter

The Stored Program

The orderly processing of data in any system or procedure normally occurs in well-defined steps. A complete procedure may generally be separated into three broad areas (Figure 91):

1. Origination of data from some source transaction, such as an order, sale, movement of goods, service, payment of wages, allocation and distribution of time, and analysis of quality.
2. Processing of the data from the source record. These operations include transcription, editing, filing, calculation, sorting, classifying, and others.
3. Preparation of results such as reports, charts, payment statements, bills, receipts, performances, and trends.

The procedure may be carried out using a great variety of methods and tools. Some steps may be clerical, particularly in the area of data origination, transcription, and editing. Where volume is large, processing may be done by machines using mechanical or electronic equipment for calculation, sorting, filing, duplicating and printing. As volume grows and complexity of processing increases, machines assume more and more importance in the system.

Once the original information has been transcribed to cards or tape, the 705 system is capable of complete processing and preparation of results. However, the procedural steps must now be defined in terms of operations that the machine can perform. Each step becomes an instruction; the series of instructions pertaining to a single procedure becomes the program (Figure 92). When the program is stored in memory, the machine can then operate upon and process the data to produce desired results.

Instructions

All 705 instructions are divided into two parts:

1. The operation to be performed: read, write, add, round, transfer, and so on. All operations are coded as a single character for storage in memory. When instructions are written, the abbreviation is normally used for convenience, rather than the machine code.
2. The operand (the address of the data in memory or the device to be selected, the number of positions to be rounded, and so on). The operand is limited to four characters.

Each instruction, therefore, occupies five positions of memory; one for the operation code, four for the operand. Instructions are also always stored in memory in such a way that the last position of the operand is located in a memory location whose address ends in 4 or 9 (Figure 93).

Programming Example

The following section describes the application of a simplified compound interest problem to the 705. It is presented to illustrate the steps normally taken in

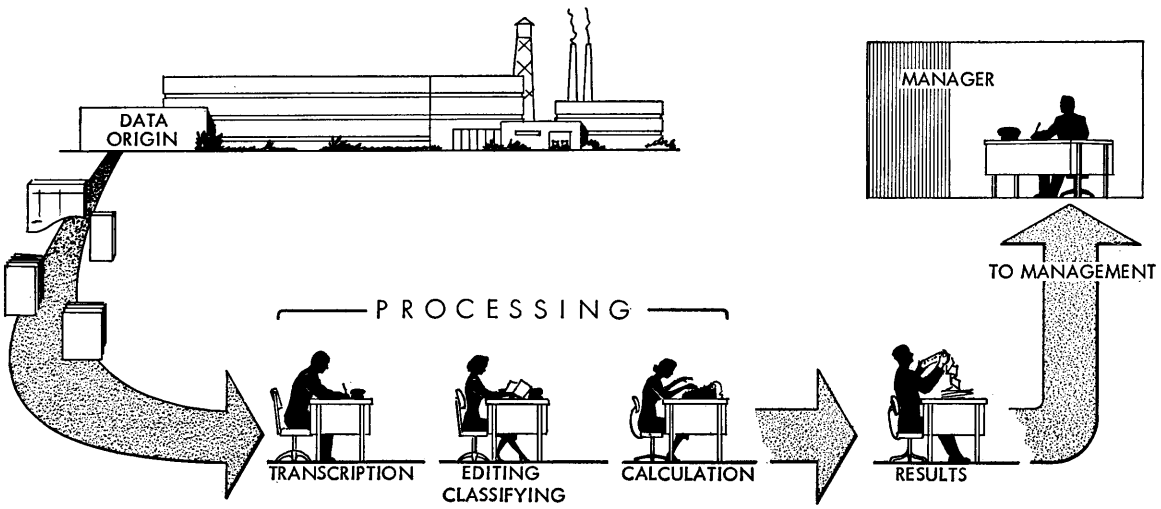


Figure 91. Schematic, Data Origin and Processing

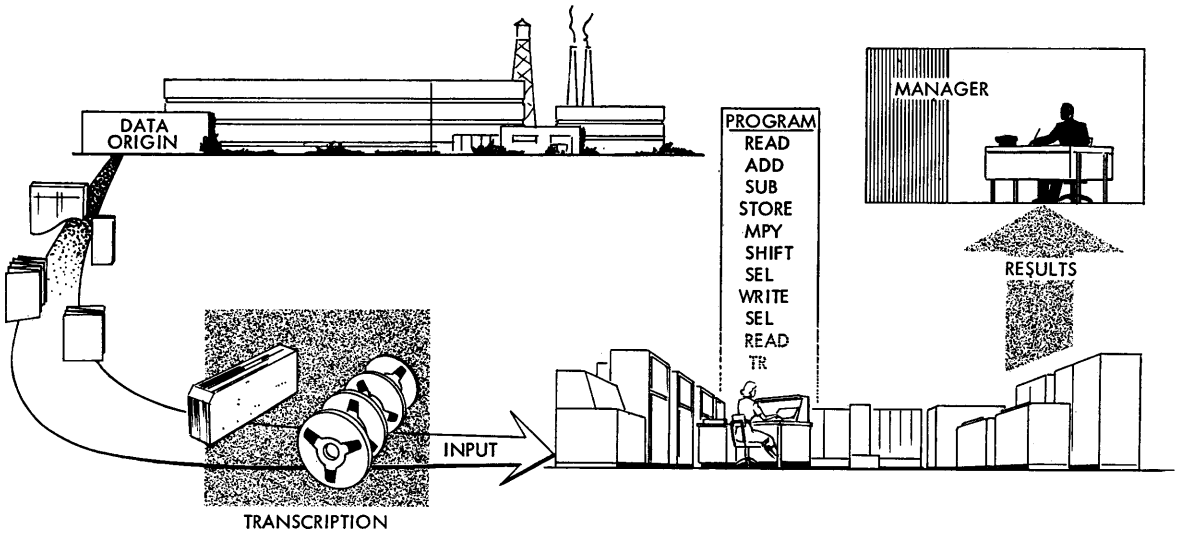


Figure 92. Schematic, Stored Program

transition from known procedures to machine methods. The problem is first analyzed in terms of this conventional procedure, a flow chart is developed, and detailed machine instructions are then written to follow the flow chart. Each instruction is briefly explained as it is used, with accompanying schematics to show how data are manipulated by the machine.

More efficient methods might be developed with the use of interest computing formulas. This could possibly be a more practical approach, if the volume of data were large enough, and the methods of computation variable. In many cases, the best approach is one based upon machine methods, rather than a program which exactly duplicates manual methods.

The completed program is shown together with instructions for loading and running the problem.

In 1627, the Indians are reported to have sold the entire island of Manhattan to the Dutch settlers for goods valued at about $24.00. Hardly anyone would argue the fact that the traders acquired one of the greatest real estate bargains of history.

An interesting problem is: What interest might the Indians have earned on the $24.00 if the money could have been placed on deposit in a savings account at

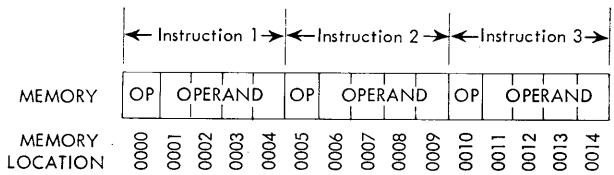


Figure 93. Instruction Location in Memory

three percent interest and compounded annually from 1627 until the present time? Admittedly, the answer is of little practical value, but the method of solving the problem on the 705 serves to illustrate the basic principles of computer programming.

First, assume that the machine can do all the required arithmetic in the same way that it might be done with paper and pencil. Using this assumption, the manual method is analyzed and a step-by-step procedure developed.

Next, this procedure is written as 705 instructions so that the machine can understand what is to be done. Finally, the resulting program of instructions and the given data are stored or "loaded" into memory for machine operation.

Figure 94 shows the paper-and-pencil approach to the problem.

The original principal of $24.00 is multiplied by the interest rate, three percent, to obtain interest for the first year. Four decimal places result. Since interest is customarily adjusted to the nearest cent, the amount is rounded two positions. Rounding is done by adding a 5 to the second digit of the interest amount and dropping the last two digits. In the example, no carry is effected until the fourth year where rounding raises the interest amount from 78 to 79 cents.

After interest is calculated and half adjusted to the nearest cent, the principal is added to produce the compounded principal. The entire calculation is then repeated for the required number of years.

While working the problem manually, it would soon become apparent that an accurate tally must be kept

First Year	$24.00	Principal
	x.03	Multiply by interest rate.
	.7200	Interest for first year
	+5	1/2 adjust second position.
	.7250	Round off two decimals.
	+24.00	Add principal.
	$24.72	Principal Plus Interest
Second Year	$24.72	Compounded Principal
	x.03	
	.7416	
	+5	
	.7466	
	+24.72	
	$25.46	
Third Year	$25.46	Compounded Principal
	x.03	
	.7638	
	+5	
	.7688	
	+25.46	
	$26.22	
Fourth Year	$26.22	Compounded Principal
	.03	
	.7866	
	+5	
	.7916	
	26.22	
	$27.01	

Figure 94. Indian Problem, Manual Method

of the number of calculations. For example, the elapsed time from 1627 to 1959 is 332 years. Therefore, the calculation is repeated 332 times.

The most convenient way for the machine to do this is to set up the quantity 332 in a counter and subtract one each time the calculation is performed. When the number in the counter reaches 0, the problem is finished.

The entire procedure may also be shown in flow chart form (Figure 95). Note that the steps, as outlined, almost exactly follow the manual method. Once instructions are set up to execute the first calculation, they can be repeated for all the following calculations. After the program has been repeated 332 times, the answer is printed and the machine stops.

Instructions must also precede the main program to read the data into memory and to set a counter to the quantity 332.

The problem is now analyzed in terms of machine operations. The next step is to reserve space in memory for storage of data. When this location is established, the instructions may be addressed to manipulate the data from the known locations. Figure 96 is a schematic showing the layout of factors for the problem.

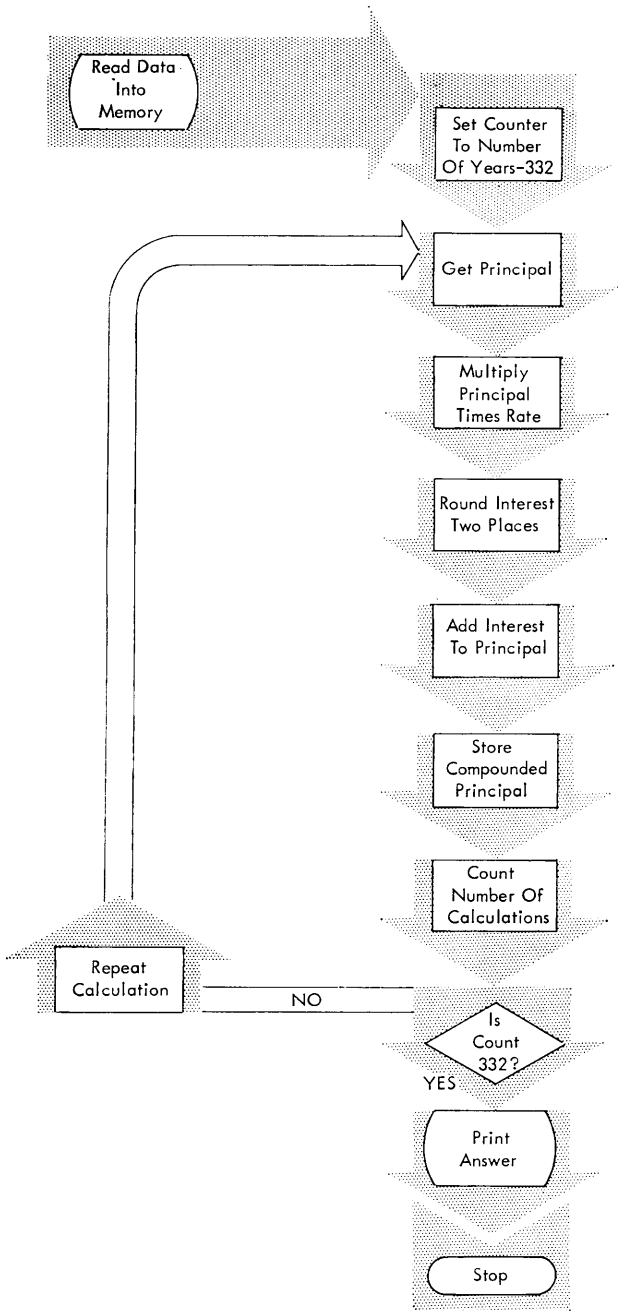


Figure 95. Flow Chart, Indian Problem

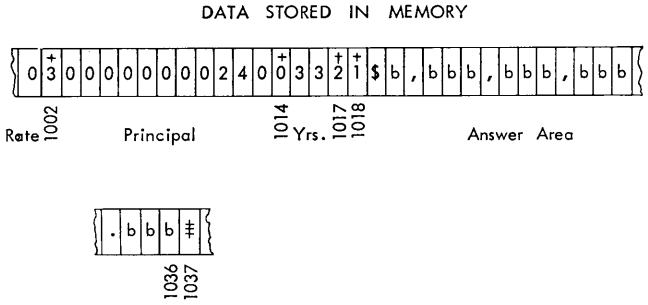


Figure 96. Data in Memory, Indian Problem

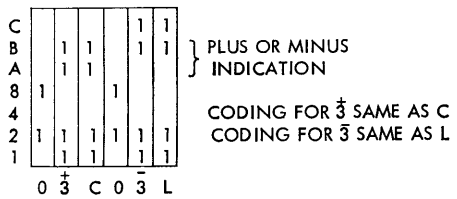


Figure 97. Signing Numerical Digits

The first factor is the interest rate, stored at memory locations 1001 and 1002 as $0\overset{+}{3}$. Note that the field is signed because it is to be used for arithmetic. In the 705, 1's in the AB positions of the character indicate plus, and a 0 in the A and a 1 in the B position indicate minus (Figure 97).

The coding of $\overset{+}{3}$ is therefore identical to coding for the letter C. The difference between $\overset{+}{3}$ and C is significant only in the manner in which the machine is instructed to use the character. As previously stated, all numerical fields are normally signed over their units position with a plus or minus indication.

The original principal amount ($24.00) is stored as a signed field in memory locations 1003-1014. This same area is also used to store the compounded principal as repeated calculations are made. Zeros are included to the left to allow enough space for the possible size of the answer. At this point, this field length is only a guess; supposedly the programmer would not take time to estimate accurately what the answer is.

The number of years of elapsed time is stored at locations 1015-1017. As each calculation is completed, a 1 is subtracted. The $\overset{+}{1}$ is stored in location 1018.

An answer area is set up in locations 1019-1036. This arrangement pre-positions the dollar sign, commas, and the decimal point for printing. All output records from the 705 must be arranged in memory exactly as they are to be printed, punched, or written on tape.

A group mark is positioned in location 1037. This special character limits the writing operation. For example, if the machine is told to write the characters on the typewriter, beginning at location 1019, the characters are typed, one at a time, through successively higher positions of memory until the group mark is sensed.

To place data in memory, the various factors are punched in an IBM card as shown in Figure 98. The data are punched in the exact sequence in which they are to be stored.

Location 0004

The first instruction of the program selects the card reader to prepare for reading the data card into memory (Figure 99). It is shown below as SEL 0100. Note that for convenience a mnemonic abbreviation represents the operation code. However, when the instruction is to be used in memory, only the single character code is stored. For SEL, this character is 2. Therefore, in memory, the complete instruction appears as 20100 and would be stored as shown in Figure 93.

Any convenient area may be used for instruction storage. Assume that, for this problem, the first instruction is stored in memory at locations 0000-0004; the second, at 0005-0009, and so on. The location is writ-

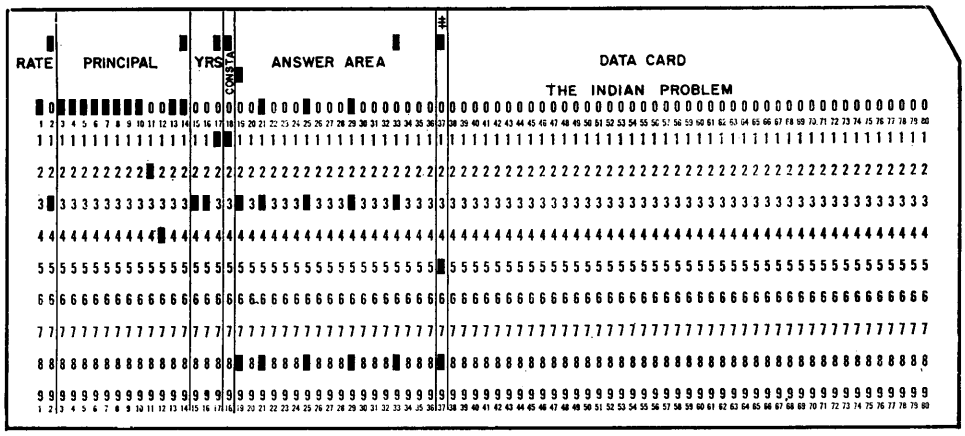


Figure 98. Data Card, Indian Problem

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader

Figure 99. Program Example

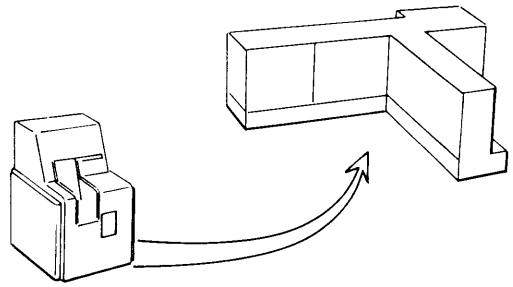


Figure 100. Select Card Reader

ten on the program sheet as 0004, 0009, 0014, and so on.

Select (2 — SEL)

The various units of the 705 system are called into use by preceding the address of the unit with a select operation. Only one unit or device may be selected at one time and the device remains selected until another select instruction is executed (Figure 100).

The select operation differentiates between a memory address and the address of a machine component. For example, the address 0100, when preceded by an ADD operation, instructs the machine to add the quantity stored at memory location 0100. When the address is preceded by a SEL operation, the instruction refers to the card reader.

Location 0009

After selection, a read instruction transfers the contents of the card reader record storage into memory beginning at address 1001 (Figure 101).

Read (Y — RD)

The read instruction is used to place data in core memory from a previously selected input device; card reader, tape unit, or drum (Figure 102).

Information is transmitted to memory starting at the location specified by the read instruction address. Transmission continues, character by character, into successively higher-order positions of memory until reading is stopped by the card reader record storage mark, an inter-record gap on tape, or a drum mark at the end of the drum record. For example, when

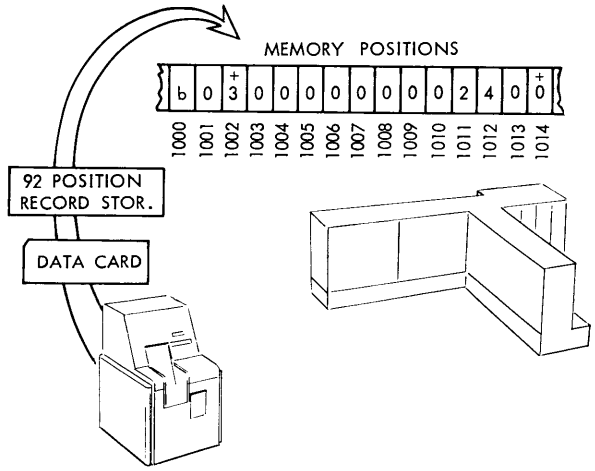


Figure 102. Read Card Record Address 1001

reading the data card shown in Figure 98, the first character is stored at memory location 1001; the second, in location 1002; the third, in 1003; and so on. Sensing the reader storage mark, wired from the control panel, terminates the read operation from a card. The storage mark is not read into memory. If no mark is wired, all 92 positions of record storage are transmitted with unwired positions as blank characters. Blank columns of the card are also converted to blank characters in memory.

The first character of a tape or drum record is stored in the memory location specified by the read address. Following characters are stored in successively higher memory positions until the end of record is sensed.

NOTE: When a data synchronizer is used, the read address must always end in 0 or 5.

Locations 0014, 0019

The number of years is set up in auxiliary storage unit 01. Each time a calculation of compounded principal is made, a 1 is subtracted from the counter. The interest rate is set up in the accumulator to prepare for calculation of interest (Figure 103).

The 705 adds, subtracts, multiplies, and divides when given arithmetic instructions. These instructions can be applied to data stored in accumulator storage, auxiliary storage (except multiply and divide) or in memory. They are normally applied to specific numerical factors or fields, such as factors developed during calculation or fields selected from records.

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card

Figure 101. Program Example

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set control ctr to no years
0019	RAD	1002		03	+			Get interest rate

Figure 103. Program Example

To select a field from memory to be acted upon by an arithmetic instruction, the field is always addressed by the memory location of its units digit. The remaining digits of the field are automatically read from right to left until a non-numerical character is reached. All characters, including blanks, are considered non-numerical except the digits 0-9. Thus, a numerical field in memory is defined as beginning at the location of its units digit and extending to, but not including, the next left non-numerical character.

Arithmetic instructions should always be addressed to "signed" fields. Both positive and negative fields in memory should be signed. Numerical fields are signed by placing a plus or minus sign indication over the units digit of the field (Figure 104). On the 705, the equivalent of the 12 zone in punched cards indicates the plus sign, and the 11 zone, the minus sign. The sign indication actually converts a numerical digit to a non-numerical character in the same manner that an 11 or 12 punch over a digit punch in IBM cards forms a letter of the alphabet (Figure 97).

The absence of a zone or the presence of a zero zone does not satisfy the requirements for a signed field. When an unsigned field is addressed by an arithmetic instruction, the sign is interpreted as plus. The correct arithmetic is performed but the machine indicates an error condition. A sign check indicator is turned on.

When an instruction calls for the movement of data between memory and storage, the address of the instruction must not only specify the location of the data, but also the storage unit to be used. Since an address is strictly limited to four characters, zone coding over the hundreds and tens positions of an arithmetic instruction address designates either accumulator storage (00) or any particular auxiliary storage unit (01-15). Figure 105 shows the codes for the accumulator and all auxiliary storage units. The codes are actually the binary numbers 1 through 15. To simplify program writing, a column on the program sheet is reserved for the number of the storage unit to be used (00 for the accumulator, 01-15 for auxiliary storage). Assume that the zone coding is given to the necessary address before the program is placed in memory.

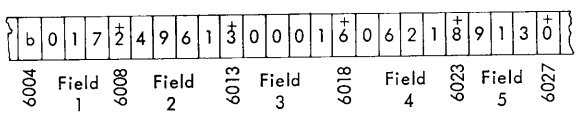


Figure 104. Signed Fields in Memory

The instruction G $\overset{01}{4}$ 3 5 0 is interpreted by the 705 and executed so that it adds the number located at position 1 4 3 5 0 to the factor in accumulator 00.

The instruction G $\overset{01}{4}$ $\overset{10}{3}$ $\overset{11}{5}$ 0 is interpreted and executed so that it adds the number at location 14350 to the factor in ASU 11.

Reset and Add (H — RAD)

The reset and add operation enters a numerical field from memory into accumulator or auxiliary storage. The address part of the instruction specifies the location of the field in memory and the storage unit to be used (Figure 106).

Digits are entered into the storage unit starting with the specified right-hand digit of the memory field. Entry continues successively from *right* to *left* (higher-order to lower-order memory locations) until a non-numerical character is sensed. The zoning (sign) of the addressed digit of the memory field is not entered into the storage unit.

The accumulator or auxiliary storage sign is set to plus when the addressed character has plus zoning, and is set to minus when the character has minus zoning. If the character has neither plus nor minus zon-

Acc. or Aux. Storage		Address			
Number	No. of Positions	Thousands	Hundreds	Tens	Units
00	256		00	00	
01	16		00	01	
02	16		00	10	
03	16		00	11	
04	16		01	00	
05	16		01	01	
06	16		01	10	
07	16		01	11	
08	16		10	00	
09	16		10	01	
10	16		10	10	
11	16		10	11	
12	16		11	00	
13	16		11	01	
14	16		11	10	
15	32		11	11	
00 indicates no zone; 01 indicates zero zone; 10 indicates 11 zone; 11 indicates 12 zone					

Figure 105. Accumulator and Auxiliary Storage Table

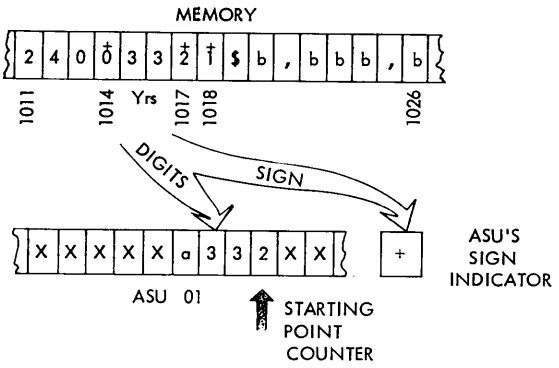


Figure 106. Reset and Add "Years" into ASU 01

ing, an error is indicated and the sign of storage is set to plus. A sign check indicator is then turned on. The sign of the designated storage is automatically set to plus when the result in storage is zero.

The field in storage is marked by the position of the starting point counter and a storage mark. The starting point counter indicates the position of the right-hand (units) digit. The storage mark is placed next to the position of the left-hand (high-order) digit of the field.

The field for number of years at location 1017 would be placed in auxiliary storage unit 01 as a332. The sign indication of all ASU's is set to plus. The field in memory is unaffected and could be placed in some other unit if desired.

Location 0024

The interest rate in accumulator is multiplied by the principal stored in memory at location 1014 (Figure 107).

Multiply (V — MPY)

The multiply instruction causes a field in memory to be multiplied by a factor in accumulator storage 00.

The multiplicand is the field in memory specified by the address part of the instruction.

The multiplier is the accumulator factor.

The product is developed in accumulator storage. The number of digits in the product is equal to the sum of the number of digits in the multiplier and

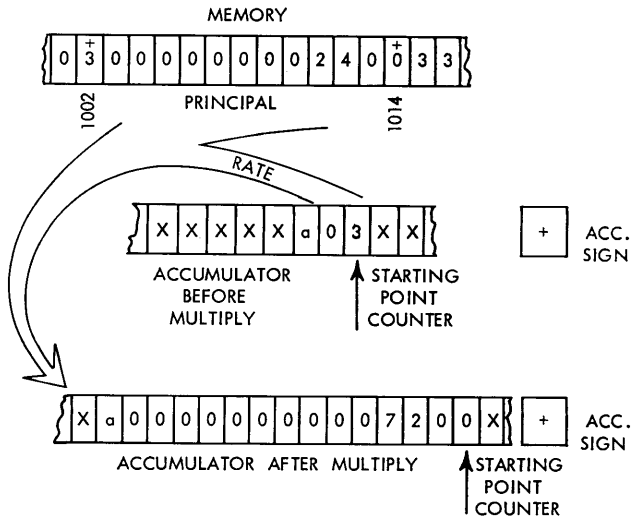


Figure 108. Multiply Rate × Principal

multiplicand. A maximum product of 128 digits can be obtained. Only the accumulator can be used for multiplication (Figure 108).

The resulting accumulator sign is plus if both multiplier and multiplicand have like signs, and minus if they have unlike signs.

Only numerical fields can be used in multiplication. The use of non-numerical fields produces inconsistent results.

When the addressed character of the field in memory has neither plus nor minus zoning, an error is indicated and the sign of the field is interpreted as plus. The sign check indicator is turned on.

Location 0029

The interest in accumulator is half adjusted to the nearest cent by a round instruction (Figure 109).

Round (E — RND)

The round instruction moves the starting point counter of the accumulator to the left the number of positions specified by the address part of the instruction. Only the accumulator can be specified. The field remaining in storage is limited to those digits between

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017				332	+	Set control ctr to no years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014		00000000007200	+			Rate x principal = interest

Figure 107. Program Example

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set control ctr to no years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014						Rate x principal = interest
0029	RND	0002						1/2 adjust interest

Figure 109. Program Example

the accumulator mark and the new position of the starting point counter (Figure 110).

A 5 is added to the digit to the right of the final position of the starting point counter. Any resulting carry is added to the units digit of the remaining storage field.

When a carry is made out of the high-order position of the original field, the result is extended one position to the left to include the carry, and the overflow check indicator is turned on.

When the result in accumulator storage is zero, the sign is always set to plus.

Location 0034

The interest in accumulator storage may now be added to the principal in memory. There are two methods of doing this. The principal may be added in the accumulator and the result stored back in memory. Two instructions are needed: one to add in storage, and a second to return the sum to memory.

In the 705, a simpler method is available. The field in the accumulator may be added directly to an arithmetic field in memory using one instruction, add to memory (Figure 111).

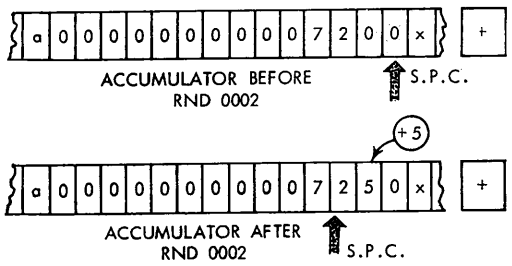


Figure 110. Round 0002

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set control ctr to no years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014						Rate x principal, = interest
0029	RND	0002						1/2 adjust interest
0034	ADM	1014						Add interest to principal

Figure 111. Program Example

Add to Memory (6 — ADM)

The add-to-memory instruction adds a field in accumulator or auxiliary storage to a field in memory. The storage unit and the memory field are specified by the address part of the instruction (Figure 112).

The result replaces the original memory field and the field in storage is unchanged.

Addition may occur in two ways depending upon whether the addressed position in memory is signed or unsigned: (1) if signed, the addition is algebraic, except that a carry beyond the next non-numerical character is lost. The proper sign is placed over the units position of the field in memory. (2) If unsigned, the addition is not algebraic. This method is explained under the section "Address Modification."

Location 0039

After each calculation of compounded principal, a count is made by subtracting 1 from the number of years (Figure 113). When the number equals 0, the problem is finished.

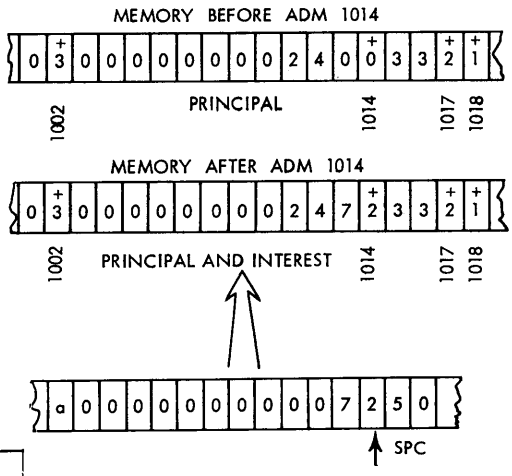


Figure 112. Add Interest to Principal

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set ctrl ctr to no years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014						Rate x principal = interest
0029	RND	0002						1/2 adjust interest
0034	ADM	1014						Add interest to principal
0039	SUB	1018	01			331	+	Subtract 1 from ctrl ctr

Figure 113. Program Example

Subtract (P — SUB)

The subtract instruction subtracts a numerical field in memory from a factor in accumulator or auxiliary storage. The address part of the instruction specifies the location of the field and the storage unit to be used (Figure 114).

Digits are subtracted from storage starting with the specified right-hand digit of the memory field and continuing from right to left until a non-numerical character is sensed. This non-numerical character is not subtracted from storage.

The result in accumulator or auxiliary storage is the difference between the storage factor and the specified memory field. The result replaces the original storage factor.

The accumulator or auxiliary storage sign is set according to the rules of algebra for subtraction. When the addressed character has neither plus nor minus zoning, an error is indicated and the sign of the field is interpreted as plus. The sign is always set to plus when the result in storage is zero.

The left-hand limit of the result is automatically set by a storage mark stored next to the highest-order digit.

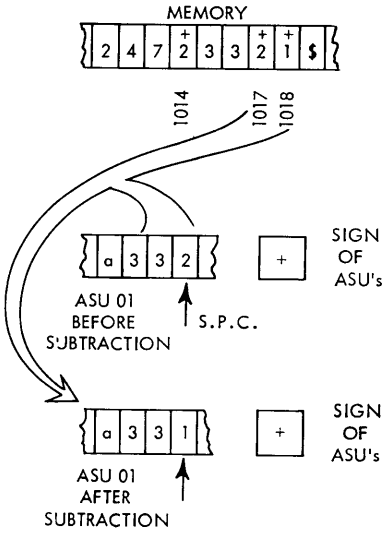


Figure 114. Subtract Operation

The length of the result equals the longer of the two fields being subtracted, unless a carry is made out of the highest-order position. In this case, the result is extended one position to include the carry as its most significant digit, the storage mark is positioned to the left of this digit, and the overflow check indicator is turned on.

When the overflow exceeds the capacity of an auxiliary storage unit, the carry is made into the adjacent unit with proper positioning of the storage mark in this unit.

The subtract instruction does not affect the field in memory.

Location 0044

The next instruction, transfer on zero specifying ASU 01, tests the contents of ASU 01 for zero (Figure 115). If the control counter equals zero, the calculation of compounded principal is completed. In this case, the address of the TRZ instruction directs the machine to location 0054 where instructions will be executed to print the answer.

If the control counter does not equal zero, the address part of the TRZ instruction is ignored and the machine proceeds to the next instruction at location 0049 in the normal manner. This instruction is an unconditional transfer (Figure 115); that is, the next instruction to be executed is always located at 0019. At 0019, the compounded principal is placed in the accumulator and the entire calculation is repeated. Subsequent instructions are repeated until the control counter is again tested for zero. The instructions at locations 0019 through 0044 form a program loop.

When the control counter reaches zero, the transfer instruction is skipped and the machine is directed to the instruction at location 0054 to print the answer.

Transfer on Zero (N — TRZ)

The transfer-on-zero instruction causes a program transfer when the zero indicator of accumulator storage or of the auxiliary storage units is turned on. The

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set ctrl ctr to no of years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014						Rate x principal = interest
0029	RND	0002						1/2 adjust interest
0034	ADM	1014						Add interest to principal
0039	SUB	1018	01			331	+	Subtract 1 from ctrl ctr
0044	TRZ	0054	01					Test ctrl ctr for zero
0049	TR	0019						Ctrl not zero, calc again

Figure 115. Program Example

accumulator zero indicator is turned on when the contents of accumulator storage consist of characters having zero numerical portions. The auxiliary storage unit's zero indicator is turned on when the contents of the last used unit consist of characters having zero numerical portions. These characters are zero, plus or minus signed zero, and the record mark.

The address part of the instruction specifies the memory location of the next instruction to be executed after the transfer. The address must also specify either accumulator storage (00) or any of the auxiliary storage units (01-15).

When a storage field consists of characters having zero numerical portions, the sign indicator is set to plus. Therefore, if a distinction is to be made between zero and plus, the transfer on zero must precede the transfer on plus.

NOTE: As a result of an incompleted division operation, the accumulator contents may be zero with the minus sign of the replaced dividend. See "Divide."

Location 0049

A transfer instruction is used to repeat the calculation if the result in ASU 01 is not zero.

Transfer (1 — TR)

The transfer instruction is used to change the sequence in which instructions of a program are executed. The address part of the instruction specifies the memory address of the right-hand digit of the next instruction to be executed.

Locations 0054, 0059

After execution of the TRZ instruction at 0044, the final answer to the problem is stored at memory location 1014 as a signed numerical field. The field is not suitable for printing, since it contains insignificant zeros and an alphabetic character in location 1014. The answer is now transferred to the memory area reserved for this purpose, the one that contains the dollar sign, commas, and the decimal point. The field is first transferred to the accumulator by a RAD instruction and then to memory by a store-for-print instruction (Figure 116).

Store for Print (5 — SPR)

The store-for-print instruction normally is used to transfer a numerical field from the accumulator or auxiliary storage to memory. However, this instruction can also be used to store alphabetic fields from the accumulator or auxiliary storage to memory.

When the sign of the storage unit is plus, a blank is stored in the memory position specified by the address part of the instruction.

When the sign of the storage unit is minus, a dash is stored in the memory position specified by the address part of the instruction. The numerical storage field is stored in the memory positions directly to the left of the sign position. The storage mark determines the left limit of the field to be stored.

When periods or commas are encountered in memory, these memory positions are skipped and the digits are stored in successively lower address positions.

Insignificant zeros, characters with zero numerical portion, and commas in the resulting field in memory

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0054	RAD	1014						Get answer into acc
0059	SPR	1036						Put answer in mem for print

Figure 116. Program Example

are replaced by blanks. The characters b, &, and – are stored as $\overset{+}{0}$ $\overset{-}{0}$, respectively. Zeros to the right of a decimal point are not replaced.

The store-for-print instruction must always be applied to fields of known lengths. For example, to store in a ten-position (plus punctuation) memory field, the storage unit must contain ten digits. If it contains less, the resulting memory field may include remaining high-order digits from a previous field.

The field in the storage unit remains unchanged by this instruction.

Locations 0064, 0069, 0074

The typewriter is selected for printing by the instruction at location 0064 (Figure 117). A write instruction prints the answer beginning at memory location 1019. The machine stops for end of job.

Write (R – WR)

The write instruction transmits a record from memory to the record storage unit of the card punch or printer. From there, it is punched in a card or printed on a report form. Records from memory are transmitted directly to the tape unit to be used for writing, to the drum section to be used for storage, or to the typewriter. The write instruction does not affect the record in memory.

Information is written from memory successively from left to right, starting at the memory position specified by the address part of the instruction and continuing until a group mark is reached. Information written is limited by the group mark.

For example, when writing on tape, the instruction WR 1201 places the character stored in memory position 1201 on the tape as the first character of the record, position 1202 as the second character, 1203 as the third, and so on until the group mark is sensed. The group mark in memory stops the writing operation and a record gap is automatically placed on the tape. The group mark is not written on tape.

When cards are being punched, the character stored in the memory position specified by the address part of the write instruction is punched in column 1. The second character from memory is punched in column 2, the third in column 3, and so on until the group mark is reached. If the length of the record is less than 80 columns, the remaining columns in the card remain unpunched. Records longer than 80 characters are punched in successive cards by a single write instruction. The write status is maintained and subsequent operations are delayed until the last block of records has been read into record storage. The 81st character of the record in memory is punched in column 1 of the second card, and so on.

On the IBM 717 Printer, the first character of the record specified in memory is printed by print wheel 1, the second by print wheel 2, and so on, until 120 characters have been written. If the length of the record is less than 120 characters, the remaining print wheels do not print. Records longer than 120 characters are printed on successive lines by a single write instruction. The write status is maintained and subsequent operations are delayed until the last block of characters has been read into record storage.

NOTE: The carriage switch on the printer may be set to PROGRAM. In this case, the first character of the record stored in memory is used for carriage control such as skipping and space control. The section "Machine Components" explains this procedure more fully.

During writing on the drum, the group mark in memory is converted to a drum mark at the end of the record.

NOTE: If a 705 III with a DS is being used, the units position of the address of the first character to be written must be a 0 or 5.

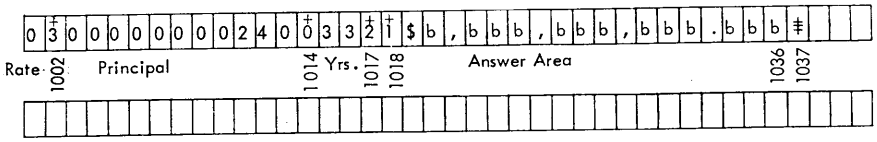
Stop (J – HLT)

Execution of this instruction stops the CPU operation. Depressing the start key causes the machine to read and execute the next instruction.

Several stops may be included in a program for the convenience of the operator. An error in reading or writing an end-of-file condition, or various other situations, may be programmed to stop or "halt" operation. The address part of the stop instruction can be read from the console when a stop occurs. The address, therefore, may be coded to indicate to the operator why machine operation has been interrupted. Any address can be given to the instruction.

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0064	SEL	0500						Select typewriter
0069	WR	1019						Write answer
0074	HLT	9999						End of job

Figure 117. Program Example



01	02	03	04	05	06	07	08
09	10	11	12	13	14	15	

INSTR. LOCATION	INSTRUCTION OPER.	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
0004	SEL	0100						Select card reader
0009	RD	1001						Read data card
0014	RAD	1017	01			332	+	Set control ctr. to no. of years
0019	RAD	1002		03	+			Get interest rate
0024	MPY	1014		00000000007200	+			Rate x principal = interest
0029	RND	0002		00000000072	+			1/2 adjust interest
0034	ADM	1014						Add interest to principal
0039	SUB	1018	01			331	+	Subtract 1 from control ctr.
0044	TRZ	0054	01					
0049	TR	0019						
0054	RAD	1014		xxxxxxxxxxx	+			
0059	SPR	1036						
0064	SEL	0500						
0069	WR	1019						
0074	HLT	9999						

Figure 118. Complete Program, Indian Problem

Complete Program

The completed program for the Indian problem is shown in Figure 118. To actually run the job, the instructions are punched in an IBM card for reading into memory. Figure 119 shows the card after punching with the instructions coded as they are to be stored.

To load instructions into memory, the operator uses the following procedure, assuming that power is on and that a card reader 0100 is available to the CPU.

1. Depress the clear memory key on the console to erase any information now in memory.

2. Place the instruction card, followed by the data card (Figure 98), in the card reader and depress the start key twice. The instruction card is now in record storage.

3. Depress the instruct key to place the machine in instruct status.

4. Key instructions as follows (705 I, II):
   a. 20100. Select card reader.
   b. Y 0000. Read instruction card beginning at memory location 0000. Instructions are read into memory locations 0000-0074 just as they are punched in the card.
   c. Depress start key. The machine executes the instruction at 0004 (the first instruction of the program) and continues automatically until the end of the job is reached.

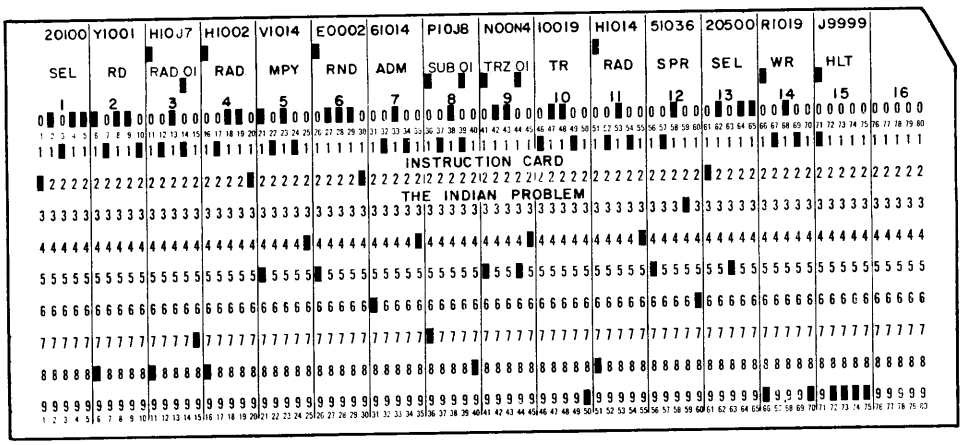


Figure 119. Instruction Card, Indian Problem

5. Key instructions as follows (705 III):
   a. Key address of card reader in memory address selector.
   b. Depress auto load key to place first card record in memory beginning at location 0000.

Summary of Additional 705 Operations

The following section summarizes the functions of operations not described previously in this "Introduction to 705 Programming" section.

Reset and Subtract (Q — RSU)

The reset and subtract instruction enters a numerical field into the accumulator or auxiliary storage from memory. The address part of the instruction specifies the location of the field and the storage unit to be used. The field in memory is not affected (Figure 120).

The accumulator or auxiliary storage sign is set to minus when the addressed character has plus zoning, and is set to plus when the addressed character has minus zoning. However, when the result is zero, the sign of storage is always set to plus.

If the addressed character is not plus or minus zoned, an error is indicated. The field in memory is assumed to be plus, thereby setting the storage sign to minus.

Add (G — ADD)

An add instruction adds a numerical field in memory to a field in accumulator or auxiliary storage. The address part of the instruction specifies the location of the memory field and the storage unit to be used. The field in memory is not affected (Figure 121).

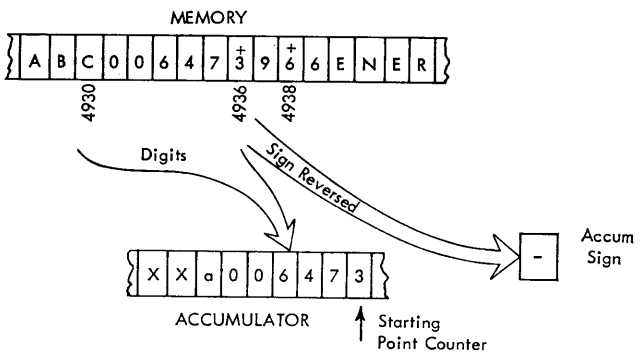


Figure 120. Reset and Subtract 4936 into Accumulator

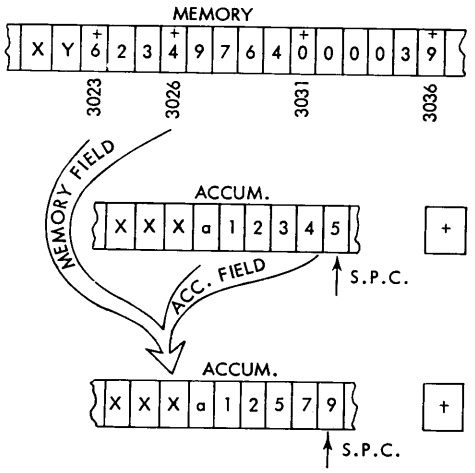


Figure 121. Add 3026 to Accumulator

The result in storage is the sum of the storage and memory fields and replaces the original contents of storage. Storage sign is automatically set according to the rules of algebra except that it is always plus when the result is zero.

If the addressed memory character does not have plus or minus zoning, an error is indicated and the memory field is assumed to be plus.

Subtract (P — SUB)

A subtract instruction subtracts a numerical field in memory from a field in accumulator or auxiliary storage. The address part of the instruction specifies the location of the memory field and the storage unit to be used. The field in memory is not affected (Figure 122).

The result in storage is the difference between the storage factor and the memory field. The result replaces the original storage factor. The storage sign is

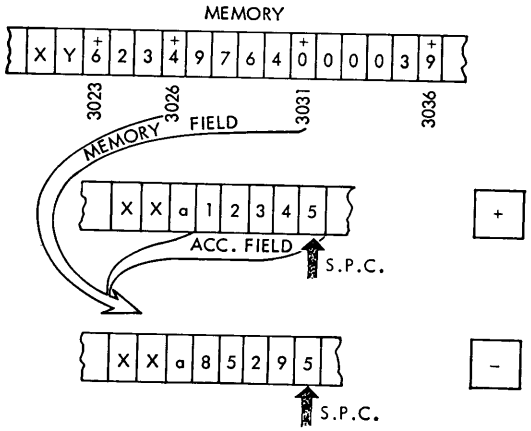


Figure 122. Subtract 3031

set according to the rules of algebra, except that it is always plus when the result is zero.

If the addressed character is not plus or minus zoned, an error is indicated and the sign of the memory field is assumed to be plus.

Store (F — ST)

The results of arithmetic operations are always obtained either in the accumulator or auxiliary storage as numerical fields. These results may be placed in memory by using a store instruction. The address of the instruction specifies the location in memory where the field is to be placed and the storage unit to be used (Figure 123).

The stored factor is automatically defined in memory as an arithmetic field. The sign of storage is placed over the units position of the field as plus or minus zoning. The next position to the left of the field in memory is examined to determine if it is a digit or non-numerical character. If it is a digit with no zoning, a plus zone is automatically added.

The factor in the accumulator or auxiliary storage is not affected and may be used for further calculation or it may be placed in other locations of memory if desired.

Divide (W — DIV)

A numerical field in accumulator storage is divided by a memory field using a divide instruction. The address specifies the location of the divisor in memory. Only accumulator storage can contain the dividend (Figure 124).

As with all arithmetic instructions, the addressed character must be signed. Otherwise, an error is indicated and the field is interpreted as plus.

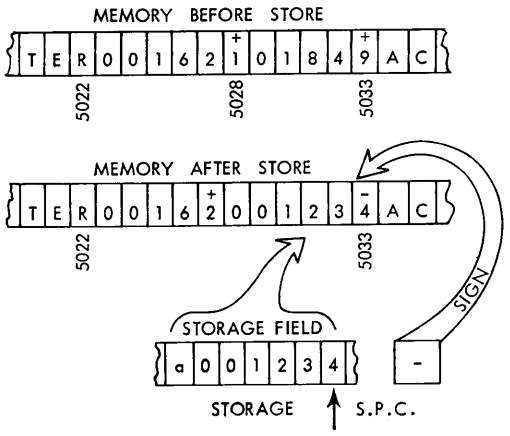


Figure 123. Store 5033

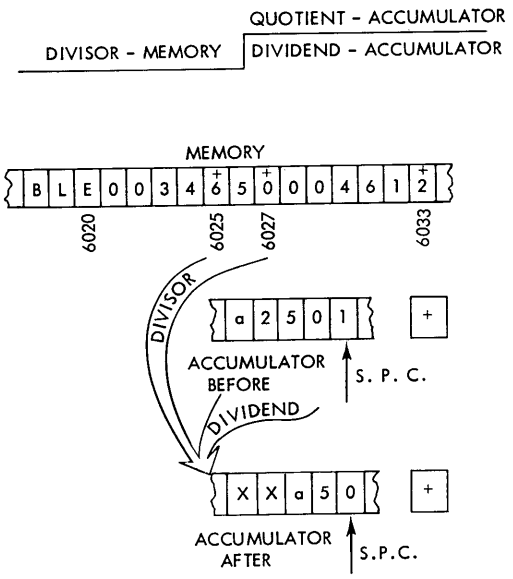


Figure 124. Divide 6027

Two additional rules apply to the divide operation:

1. The dividend must contain a greater number of digits than the divisor.
2. The divisor must be greater in value (regardless of sign) than an equal number of digits at the left end of the dividend.

If these rules are violated, an error is indicated and the machine proceeds to the next instruction. Since the magnitude and length of fields used in division are normally known to the programmer, the dividend may be adjusted by adding zeros to the left to prevent violation of the divide rules. If the conditions are not always known, an error condition may branch the program to instructions that take corrective action.

The accumulator sign is plus if the divisor and dividend have like signs, and minus if they have unlike signs.

Set Left (B — SET)

The set left instruction adjusts the length of the accumulator or auxiliary storage field to the number of characters specified by the address part of the instruction. The adjustment is made to the left end of the field by properly placing the storage mark (Figure 125).

If the mark is placed beyond the left position of the field, the storage positions between the field and the mark are filled with zeros. If the mark is placed in some position of the field itself, that position of the field and any positions to the left of the mark are no longer considered as part of the storage contents.

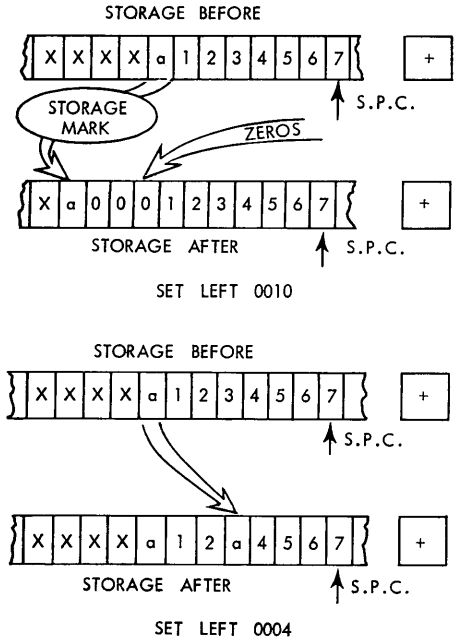


Figure 125. Set Left 0004

Therefore, the positioning of the storage mark either extends the field to the left and adds zeros, or it cuts off significant digits from the left end of the field.

The set left instruction may also be used to couple auxiliary storage units. For example, SET 0018 specifying ASU 01 extends ASU 01 to 18 positions. Sixteen positions of the field are located in ASU 01; two positions, in ASU 02.

Lengthen (D — LNG)

The lengthen instruction shifts the starting point counter of the accumulator to the right. The address of the instruction specifies the number of positions to be moved (Figure 126).

A zero is added to the right of the accumulator field for each position moved by the counter.

The lengthen instruction may only be used with accumulator.

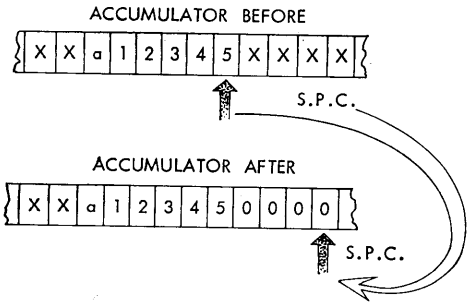


Figure 126. LNG 0004

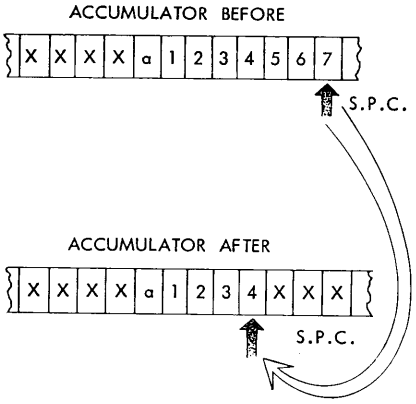


Figure 127. SHR 0003

Shorten (C — SHR)

The shorten instruction shifts the starting point counter of accumulator storage to the left. The address of the instruction specifies the number of positions to be moved (Figure 127).

Because the field in storage consists of those characters betwen the position of the starting point counter and the storage mark, the movement of the counter to the left has the effect of removing characters from the right end of the storage field.

The shorten instruction can only be used with accumulator storage.

Transfer on Zero (N — TRZ)

The transfer-on-zero instruction causes a program transfer when the zero indicator of accumulator storage or of auxiliary storage is turned on (Figure 128).

The address part of the instruction specifies the memory location of the next instruction to be executed. The address must also specify either accumulator or a particular ASU. Auxiliary storage zero indi-

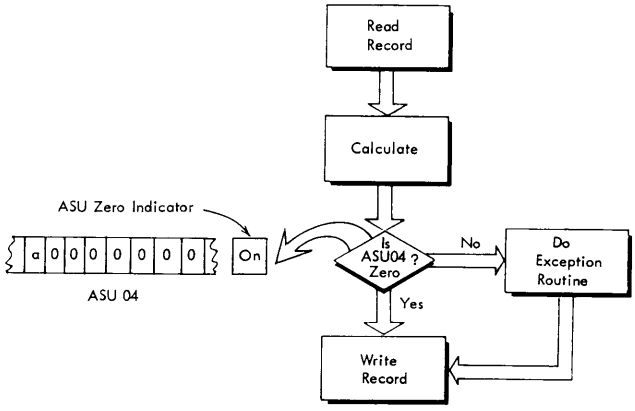


Figure 128. Transfer on Zero

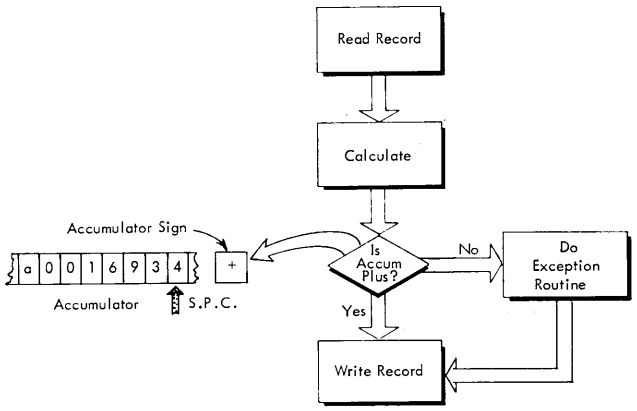


Figure 129. Transfer on Plus

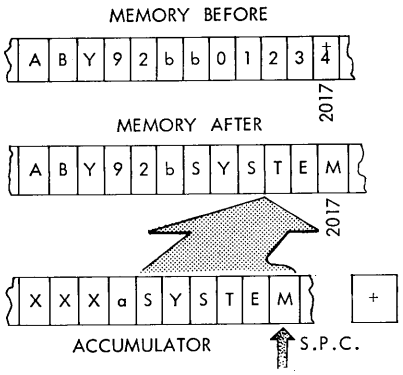


Figure 131. UNL 2017

cator is set when the contents of the unit *last used* are zero.

Transfer on Plus (M — TRP)

The transfer-on-plus instruction causes a program transfer when the sign of accumulator or auxiliary storage is plus. Figure 129 shows a program testing the sign of the accumulator. The address part of the instruction specifies the location in memory of the next instruction to be executed. The address must also specify either accumulator or a particular ASU.

Note that the sign of auxiliary storage is set to plus when the result in the last used ASU is plus.

Load Storage (8 — LOD)

The load instruction places a designated field in memory in either the accumulator or an auxiliary storage unit. The size of the field is determined by the positioning of the starting point counter and the storage mark which adjusts the length of the contents of storage (Figure 130).

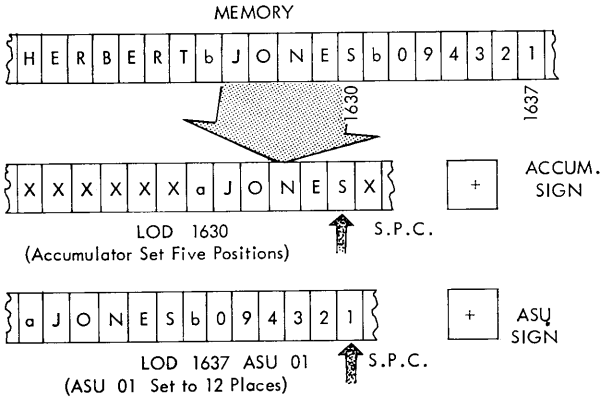


Figure 130. Load 1630

For example, if a set left instruction adjusts storage to ten positions, a following load instruction for that storage unit loads ten characters from memory. A memory area of any size may be loaded up to the capacity of storage.

The address part of the instruction specifies the right-hand memory location of the memory data and the storage unit to be used.

The storage sign is always set to plus by the load instruction. The contents of memory are not affected. Characters are placed in storage exactly as stored in memory with no regard for the sign of a numerical field. Thus, storage may be used to hold alphabetic, numerical or mixed fields as required.

Unload Storage (7 — UNL)

The unload instruction accomplishes the reverse of load. Information in storage is placed in memory at the location specified by the address part. Any storage unit or accumulator can be specified (Figure 131).

The number of characters placed in memory is determined by the number of positions available as the contents of storage. Therefore, if the storage mark is ten positions to the left of the starting point counter, ten characters are placed in memory.

The storage sign has no effect upon the data placed in memory. Contents of storage remain the same.

Compare (4 — CMP)

The compare instruction compares the contents of accumulator or auxiliary storage with a portion of memory specified by the address of the instruction. The particular storage unit to be used is also designated by the address.

The contents of storage are compared with an equal number of positions of memory. Comparison proceeds

in the usual way; that is, the most significant characters are those on the left.

All characters that can appear in memory can be compared, including record mark and group mark. The ascending sequence of 705 characters is as follows:

blank . ⌑ ≠ & $ * – / , % # @ $\overset{+}{0}$ A through I $\overline{0}$ J through R ≠ S through Z 0 through 9

Transfer on High (K — TRH)

When a comparison determines that the storage field is higher than the field in memory, a high indicator is turned on. The indicator may be interrogated by a following transfer-on-high instruction. If the indicator is on, the next instruction to be executed is located at the address specified by the transfer instruction. If the indicator is off, no transfer is effected.

The indicator remains on until another comparison is made and may be interrogated as many times as desired.

Transfer on Equal (L — TRE)

The transfer-on-equal instruction executes a program transfer if the results of a comparison are equal. The location of the next instruction to be executed is specified by the address of the transfer. If the comparison is not equal, no transfer is executed. The transfer-on-equal instruction can be used any number of times between comparisons.

When comparison between storage and memory determines that the storage field is lower than memory, neither a transfer-on-equal nor a transfer-on-high instruction can be executed. The next instruction in sequence is performed in the normal manner. Figure 132 shows how the program may be branched as a result of a comparison.

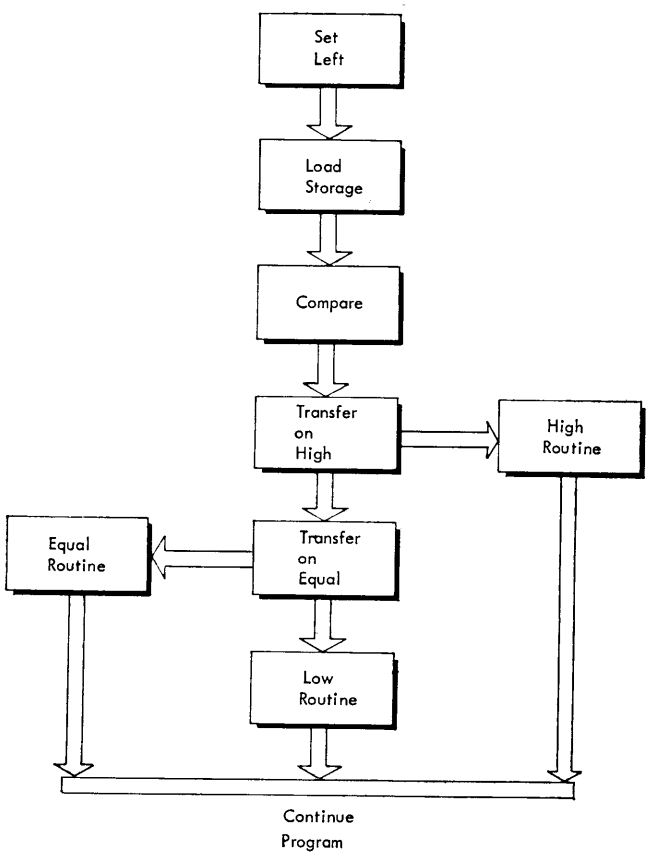


Figure 132. Branching after Comparison

Data Transmission

Various methods may be used to move data in the 705 from one memory location to another. These methods provide ability to transfer records from input to output areas, rearrange and combine records, put information in sequence and classify it, and work with any portion of the record or records in memory. Specific operations are available to the programmer to move data from memory to memory, from memory to storage, or from storage to memory. Data to be transmitted from one storage unit to another, however, must always pass through memory. The transmission may specify particular fields, groups of fields, or individual characters.

Transmission may be character-by-character or in groups of five characters.

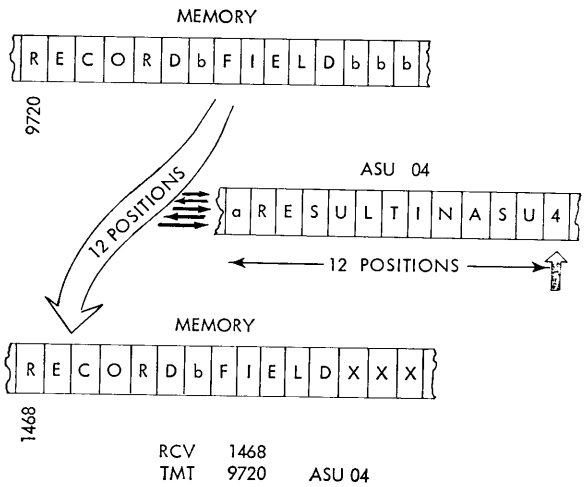


Figure 133. Receive-and-Transmit Operation, Serial

Receive Serial (U — RCV)

A receive instruction is used to specify a location in memory to which information is to be transmitted from some other portion of memory. This location is specified by the address of the instruction (Figure 133).

The address, like that of a read or write address, specifies the first left-hand character of the group of characters to be moved.

Because two portions of memory are involved in the transmission, two counter or indexing devices are used. The address of the receive instruction is placed in memory address counter II (MAC II). The counter steps up one position for each character moved.

Memory address counter (MAC I) contains the address of the characters to be transmitted as explained below.

Transmit Serial (9 — TMT)

The transmit instruction address specifies a location in memory from which information is to be placed in another area of memory. It normally follows a receive instruction. The address also specifies an auxiliary storage unit and refers to the memory location of the first character to be transmitted.

The number of characters sent to the receive area is determined by the number of positions between the starting point counter and the storage mark of the specified ASU. That is, if transmission is controlled by ASU 10 set to 12 places, 12 characters are sent to the receive area. Any ASU may be specified. Neither the storage unit nor its contents is affected by the transmit instruction. The unit may be previously adjusted by a set left instruction or it may be set properly as a result of calculation.

Memory address counter I is set by the address of the transmit instruction and steps one for each character transmitted.

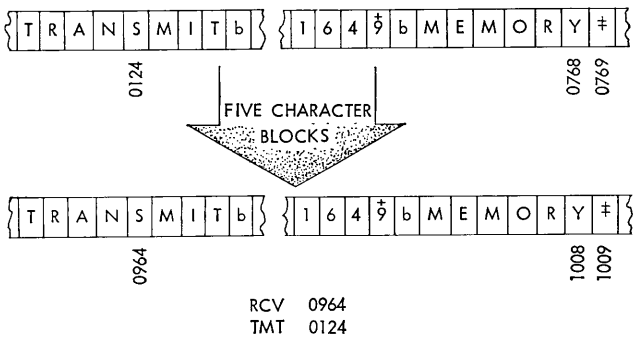


Figure 134. Receive-and-Transmit Operation, Five-Character

Receive (U — RCV)

Five-character transmission may also be used in a manner similar to single-character transmission. In this case, blocks of five characters are received. The address of the instruction specifies the location of the fifth character of the first block. The address must always specify the memory location with an address ending in 4 or 9 (Figure 134).

The address of the receive instruction is also placed in MAC II which steps five positions for each block of data received.

Transmit (9 — TMT)

The transmit instruction normally follows a RCV. When no ASU is specified, a five-character transmission occurs.

The address of the TMT designates the units position of the first block of five characters to be transmitted to the receive area. The address must always refer to a memory location ending in 4 or 9 and the total number of characters affected is always evenly divisible by 5.

Transmission is limited by a record mark in the units position of the last block of five characters. Sensing the mark terminates the operation and the record mark is moved as any other character. The mark must therefore be placed in a memory position ending in 4 or 9.

In any other location, the transmission is not affected.

The address of the TMT is placed in MAC I which steps five characters at a time as transmission occurs.

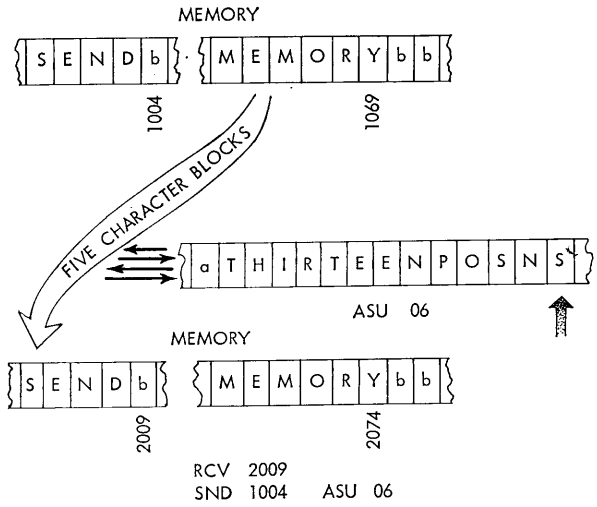


Figure 135. Receive-and-Send Operation

Send (/ — SND) *(705 III)*

The send instruction is similar in operation to the five-character RCV and TMT (Figure 135). It causes high-speed movement of data from one location in memory to another. However, it is not necessary to place a record mark to limit transmission. Instead, the number of five-character blocks sent is controlled by a preset ASU.

As in the transmit operation, the SND must be preceded by a RCV instruction to set MAC II at the location of the receiving area. Both RCV and SND instructions must have addresses ending in 4 or 9. The address of both instructions refers to the units position of the first block of five characters to be moved.

Five characters in memory are moved for each single position of the designated storage unit. Contents of storage and the transmit area of memory are not affected.

Blank Memory ($ — BLM) *(705 III)*

The blank memory instruction is used for such operations as the blanking of an output area before arranging the next record for printing or punching. BLM may be executed in two ways (Figure 136).

1. The address of the instruction, with no ASU specified, indicates the number of five-character blocks to be filled with blanks. A preceding RCV instruction with an address ending in 4 or 9 specifies the receive area. Example:

```
RCV   25024
BLM    0003
```

These instructions cause blanking of 3 groups of five characters, beginning with the character at location 25020 and ending in location 25034.

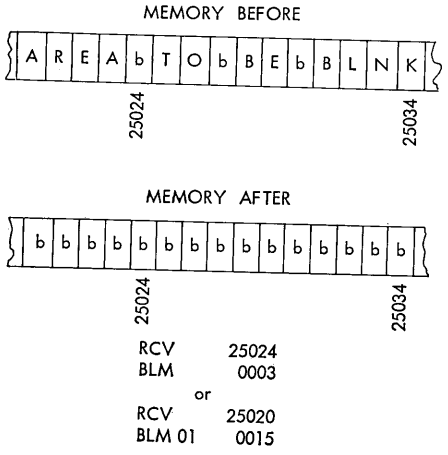


Figure 136. Receive-and-Blank Operation

2. The address of the instruction may specify ASU 01 (and only ASU 01). In this case, individual positions of memory will be blanked. The number of blanks generated is controlled by the numerical portion of the BLM address. A preceding RCV instruction specifies the address of the character in memory where blanking will begin. Example:

```
RCV     25020
BLM 01   0015
```

These instructions cause the blanking of a total of 15 characters, starting with location 25020 and ending with location 25034.

Read while Writing (S — RWW)

A record or group of records may be read into memory from tape while at the same time another block of data may be written from memory onto tape. As a result, the tape time may be overlapped. Two areas

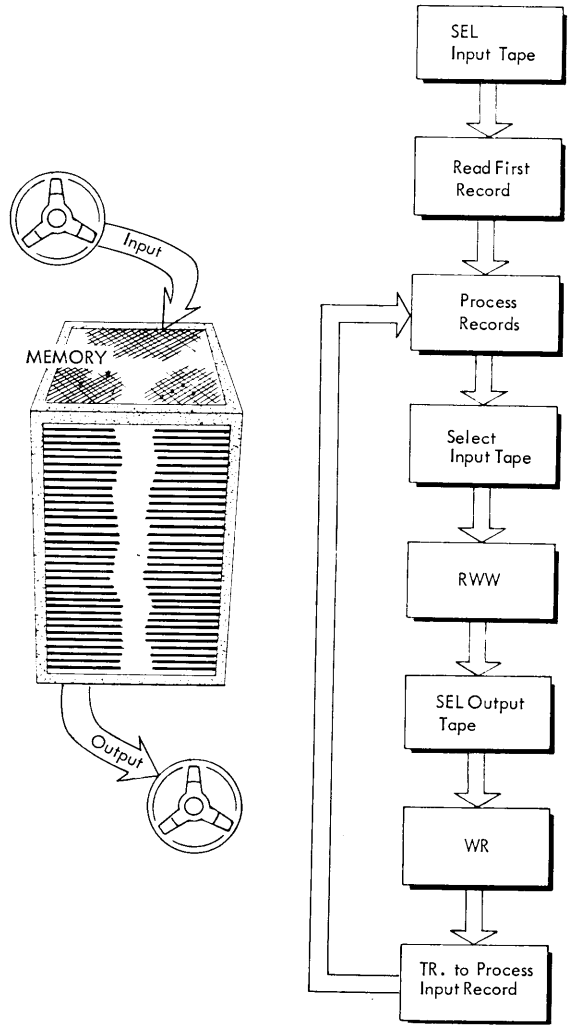


Figure 137. Read-while-Writing Operation

of memory must be set aside for this operation, one for reading in and the other for writing (Figure 137).

The read-while-writing instruction conditions a previously selected input tape unit to retain its selected status. The instruction can only be used with an IBM 754 Tape Control. It prepares the unit to read, but no reading is done until a subsequent write instruction is executed. The address part of the instruction specifies the memory location of the first character to be read in. This address sets MAC II.

After the output tape is selected, a write instruction causes simultaneous reading and writing. The write address is the location of the first character to be written. The address also sets MAC I.

Zoning of Data Fields

A number of operations in the 705 permit the programmer to manipulate the numerical and zone portions of data separately. Such instructions are useful to adjust the sign of numerical fields, to act upon operation codes as data or to examine and change any individual binary bit within a character in any position of memory.

The following examples illustrate the principles of manipulating zones in order to sign fields from incoming IBM card records and to change an operation code for the purpose of constructing a program switch.

The numerical fields in cards are commonly signed by punching a specific zone punch over the units digit of the field to denote minus, or with no zone punching to indicate plus. When fields of this kind are read directly into memory, both the plus and minus fields must be given proper zoning to define them as numerical fields. Assume that one six-position card field is in memory at location 4168. Position 4168 may contain either no zoning or minus zoning.

If no zone is present, signifying plus, the SGN 4168 generates a plus in the accumulator which is placed in the same location by ADM 4168. If the zone is minus, the minus zone is removed by the SGN instruction, and replaced by the ADM instruction.

The same instructions will also properly sign the fields in memory if the X punch designating minus is placed in some other position of the field, for example, location 4163. In this case, the address of the sign instruction becomes 4163 and the ADM instruction remains 4168.

Sign (T — SGN)

The sign instruction is used to remove any zone from a single memory character and place it in the accumulator or auxiliary storage. The character affected and the storage unit to be used are specified by the address of the instruction.

When the zoning of the addressed character in memory is minus, a dash (minus zone) character is placed in the storage unit and the storage sign is set to minus.

When the zoning of the addressed character is other than minus or no zone, an ampersand (plus zone) is placed in storage with the storage sign set to plus.

The addressed character remains in memory with no zoning.

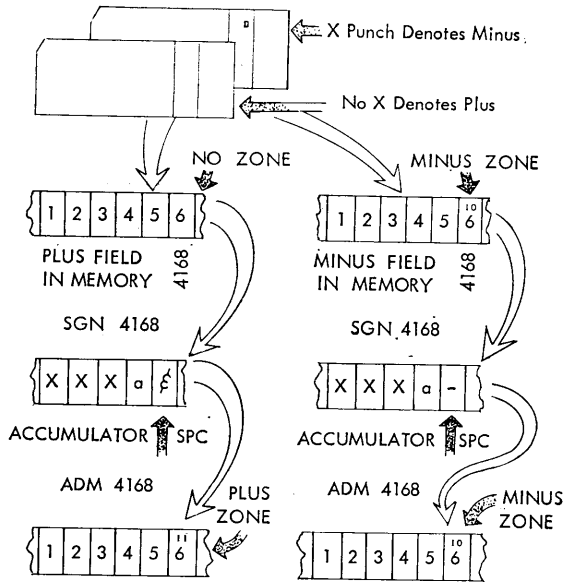


Figure 138. Signing Fields in Memory

The sign in storage (ampersand or dash) may be given to any memory character by an add-to-memory instruction (Figures 138 and 139).

Add to Memory, Unsigned Fields (6 — ADM)

The add-to-memory instruction may be addressed to an unsigned field in memory. In this case, the addition is not algebraic. It begins with the right-hand

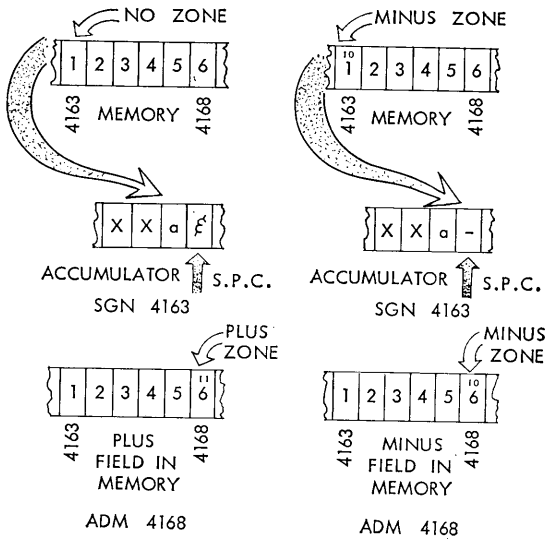


Figure 139. Signing Fields in Memory

digit of accumulator or auxiliary storage and the addressed character in memory and continues from right to left until the storage mark is reached.

If non-numerical characters, including blanks, are encountered in either memory or storage, both zones and digits are added separately. The zone portions are added as binary numbers; the numerical parts are added decimally.

Carries are propagated binarily or numerically through each portion of the memory field, except that any carry beyond the position of the storage mark is lost. The contents of storage are not affected.

Branching

The flow chart shown in Figure 140 illustrates a simplified routine for file maintenance. In such a procedure, a master file is normally adjusted by current transactions to bring it up to the latest accounting period. These two files, master and detail, become input to the 705 system.

As the master records are read in, they are normally in ascending sequence by some key field, account number, customer number, part number, and so on. The detail records must be in corresponding sequence by the same key field. As each detail record is read in, it must be compared against the master to ascertain if it applies to the particular record just read into memory. Normally, detail records do not apply to some masters; the ratio of activity to the file varies, depending upon the particular procedure involved.

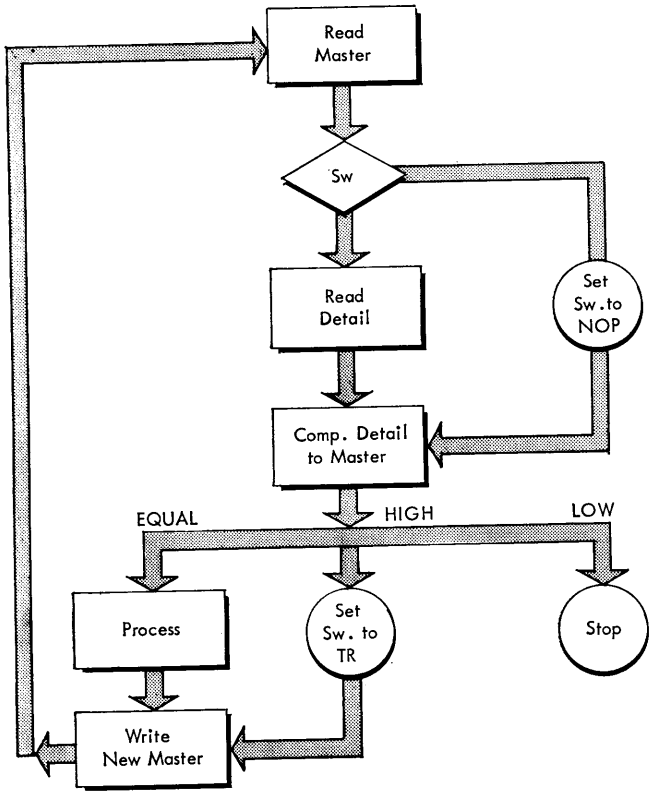


Figure 140. File Maintenance, Basic Flow Chart

If the detail matches the master, the results of the comparison are equal. The program should then follow the instructions to process the detail against the master.

If the detail record is higher in the key sequence than the master, it indicates that the master record now in memory has no corresponding detail. It is considered inactive, and is to be merely copied on the output file without processing. The result of a high comparison, therefore, must direct the program to a series of instructions that accomplish this.

If the detail record is lower than the master, it signifies that the detail does not match and cannot be processed. This is normally an error condition where the unmatched record must be written out separately as unmatched. In the flow chart, this branch of the program indicates a machine stop.

The entire operation may be compared to the function of a collator matching and merging two files. However, the 705 executes its instructions in sequence and, in order to feed the master file without feeding a detail, the instructions for reading details must be skipped.

To accomplish this, a program "switch" is inserted between the instructions for reading masters and reading details. The switch may be set by the machine, depending upon which condition of high, equal, or low comparison is encountered.

An unconditional transfer instruction is placed in the program as a switch. When the operation code of this instruction (a digit 1) is executed, a transfer is made directly to the comparison instruction. In this condition, the switch is closed.

To open the switch and allow the program to ignore the transfer, the operation code 1 is changed to A. The character A is the code for "no operation." That is, the machine does not execute this instruction but takes the next instruction in sequence.

The SGN and ADM instructions are used to set the switch to either no operation or transfer. The SGN instruction, addressed to the switch, removes the zoning from the character A, converting it to a digit 1. The ADM replaces the zoning, changing the 1 to a character A. Figure 141 shows the essential steps of the program written in flow chart form.

The ability of the 705 to operate upon instructions as data is one of the most important aspects of the

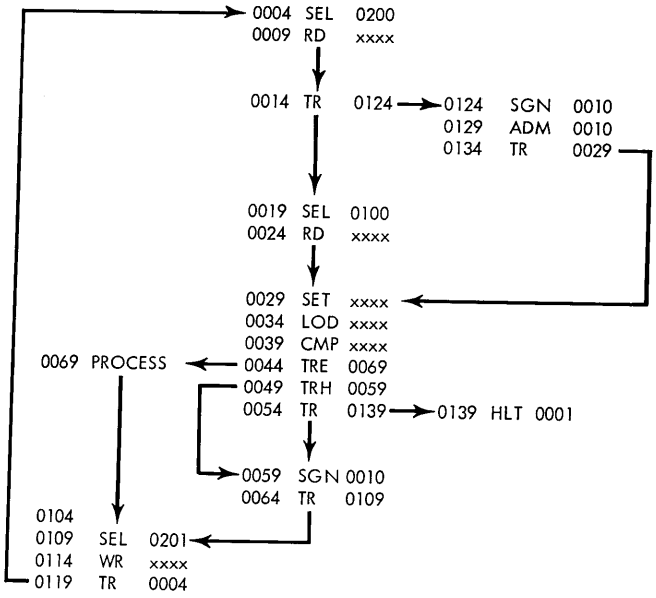


Figure 141. File Maintenance, Program Schematic

stored program. Among other advantages, it enables the machine to adjust its own procedure depending upon the varying conditions of the problem to be solved. Not only may the operation codes be changed by instructions, but addresses may be changed as well. Thus, a single instruction may perform the work of many with a consequent saving in memory space.

Address Modification

In general, address modification serves two purposes.

1. The number of instructions in a program may be reduced, conserving memory for data or other storage.
2. The machine may vary its own instructions depending upon circumstances encountered during the working of a procedure. A basic flow of work controlled by the program can thus serve as a pattern of procedure which can change as required by the type of entry data, the result of calculation, various error conditions, end-of-file detection, and so on.

The ability of the machine to use limited judgment while it performs work adds greatly to its flexibility and the extent of its application. The degree of judgment exercised is written into the program within the framework of the possible operations which it can execute. The efficient and practical use of this judgment is directly related to the skill and ingenuity of the programmer. Address modification is one means of making the program more effective.

The programmer should always keep in mind, however, that while the total number of instructions may be reduced by modification, the number of machine operations may be increased. In many cases, additional instructions may be more efficient, provided the necessary memory area is available. These conditions vary with each application of the 705.

As an example, the address part of instructions selecting the various components of the 705 system may be modified by other instructions in the program. One use of this type of modification is the selection of alternate tape units when an end-of-file 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.

In the program shown in Figure 142, two tape units are used alternately to read a file. Addresses of the two units are 0201 and 0203.

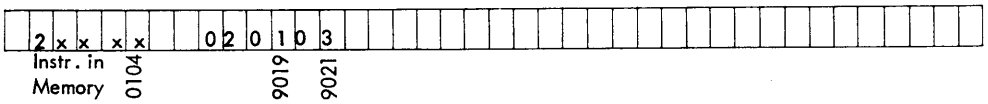
The units and tens positions of each tape address are stored as constants in memory at locations 9019 and 9021. The select address in the program for the input tape is reset to 0201 and the constants 01 and 03 are loaded into auxiliary storage units 01 and 02 by preliminary instructions.

When a transfer on signal is effected by an end of file, the contents of ASU 01 are compared against the address part of the tape unit select instruction at location 0104. When the address is 0201, that is, when the comparison is equal, ASU 02 is unloaded, changing the address to 0203. When the address is 0203, the comparison is unequal and ASU 01 is unloaded changing the address to 0201. The select address therefore alternates between address 0201 and 0203.

Figure 143 illustrates a second method of alternating between addresses by calculation. The sum of the units and tens position of two tape addresses is stored in memory as a constant factor. Any two component addresses may be alternated by a two-digit factor, if the sum of their units and tens positions is not greater than 99. In Figure 143, tape unit addresses 0201 and 0203 are used. The factor in memory is therefore the sum of 01 and 03 or 04, and is stored at location 9021.

When end of file is signaled from tape unit 0201, the units and tens position of the tape select instruction (location 0104) are loaded in ASU 01. The constant 04 is subtracted to obtain a minus result. This result is unloaded into the tape address. The minus sign of the result is ignored by unloading.

When end of file is signaled from tape 0203, the constant 04 is subtracted from 03 to obtain the re-



INSTR LOCATION	INSTRUCTION OPER	INSTRUCTION ADDRESS	STOR. CODE	ACCUMULATOR 00	SIGN	AUXILIARY STOR. REG. 01-15	SIGN	EXPLANATION
	Housekeeping							
0004	SET	0004	01			axxxx		
0009	LOD	9019	01			a0201	+	
0014	UNL	0104	01					Reset Select Address
0019	SET	0002	01			a01	+	Get Constant 01
0024	SET	0002	02			axx		
0029	LOD	9021	02			a03	+	Get Constant 03
	Main Routine							
0104	SEL	[0201]						Input Tape Unit
0109	RD	xxxx						Read Record
0114	TRS	1014						End of File
	Change Address							
1014	CMP	0104	01					A.S.U. 01 vs. Select Address
1019	TRE	1034						Address is 0201
1024	UNL	0104	01					Change Address to 0201
1029	TR	0104						
1034	UNL	0104	02					Change Address to 0203
1039	TR	0104						

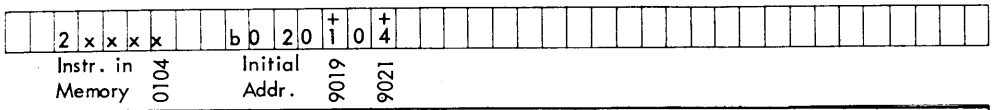
Figure 142. Program, Alternate Tape Units

sult 01. Unloading this result changes the tape address from 0203 to 0201. The select address alternates between 0201 and 0203 each time end of file is signaled.

The address part of instructions specifying memory locations from 0000 to 9999 may also be modified in either of two ways: (1) by arithmetic operations in the storage units, or (2) by an add-to-memory instruction. These methods can be used in the 705 I and II only if the result of the modification never produces a carry beyond the four digits for the lower memory addresses. Also, if the address to be modified is an arithmetic one, such an address can specify only the accumulator. This is because any zoning used to specify ASU's is removed by calculation. Again, this limitation applies only to the 705 I and II. The 705 III uses special operation codes to deal with such addresses as will be explained in following sections.

The address part of an instruction contains only four characters. To specify locations above 10,000 (actually a five-digit address), zoning is placed over the high-order position of the address. In the 705 III, zoning is also placed over the units position. The zoning substitutes for the fifth or ten-thousandth position.

Addresses with zoning may be modified by using add-to-memory to unsigned fields. This operation alters



INSTR LOCATION	INSTRUCTION OPER	INSTRUCTION ADDRESS	STOR CODE	ACCUMULATOR 00	SIGN	AUXILIARY STORAGE 01-15	SIGN	EXPLANATION
	Housekeeping							
0004	RAD	9019	01			a0201	+	
0009	UNL	0104	01					Reset Select Address
0014	SET	0002	01					
	Main Routine							
0104	SEL	[0201]						Input Tape Unit
0109	RD	xxxx						Read Record
0114	TRS	1014						End of File
	Change Address							
1014	LOD	0104	01			a01	+	Get Two Positions of Address
1019	SUB	9021	01			a03	−	Subtract Constant 04
1024	UNL	0104	01					Change Select Address
1029	TR	0104						

Figure 143. Program, Alternate Tape Units

the numerical portion of the address without disturbing any zoning. Conversely, the zoning portion may also be modified without changing the numerical portion.

In the 705 III, the possibility of using a machine with a memory capacity of as many as 80,000 characters means that many, if not most, of the instructions in a program refer to locations over 10,000. Thus, these addresses have zoning of the thousands and units positions. This increased use of zoning not only prevents considering an address always as an unsigned field in memory but also may tend to complicate the manipulation, modification, and comparison of instruction addresses.

Three operations available in the 705 III are designed to operate on addresses rather than general data. They must always be addressed to a memory field ending in 4 or 9; otherwise, an instruction check indicator is turned on.

Load Address (# — LDA) *(705 III)*

The load address instruction is used to load the address portion of an instruction in memory into the accumulator or auxiliary storage as a five-digit field (Figure 144).

Only the numerical portion of the four addressed characters in memory is loaded into storage, forming the four low-order digits there. The fifth (high-order) digit in storage is formed from the zone bits over the units and thousands positions of the address in memory.

All other zoning in the memory field is ignored, including any ASU zoning over the tens and hundreds positions.

The resulting five-digit field in storage is automatically defined by a storage mark. Therefore, LDA need not be preceded by a SET instruction.

Unload Address (— ULA)* *(705 III)*

The unload address instruction is the reverse of the load address instruction. It causes a five-digit field in storage to be placed in memory as a four-character address. All other zoning in the memory field is undisturbed (Figure 145).

If the field in storage is less than five digits, it is placed in memory as four digits with zeros to the left. If the field in storage is longer than five digits, only the low-order five are placed in memory.

Add to Address in Memory (@ — AAM) *(705 III)*

This instruction adds a five-digit field in storage to a four-character field in memory. The first four low-order digits in storage are added to the numerical portion of the four characters of the address in memory. The fifth (high-order) position in storage is added to the address zones. The result appears as a proper four-character address in memory (Figure 146).

All additions with an AAM instruction is non-algebraic, always adding the absolute value of the amount in storage, regardless of sign.

If the field in storage is greater than five digits, the instruction adds only the first five digits (low-order) and ignores any others. The field in storage may also be less than five digits.

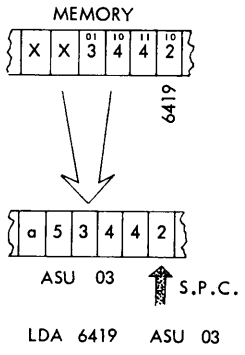


Figure 144. Load Address Operation

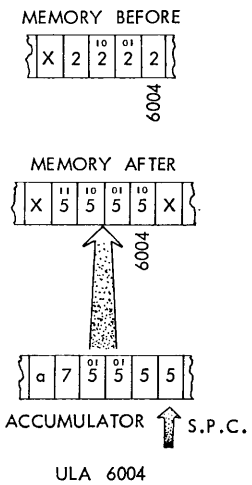


Figure 145. Unload Address Operation

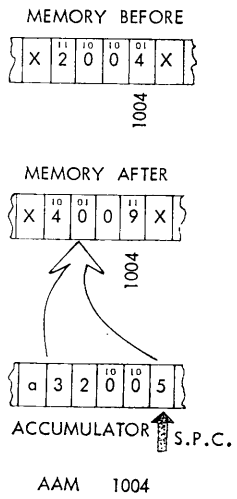


Figure 146. Add-Address-to-Memory Operation

Bit Manipulation

The 705 III can perform other operations, not performed by the 705 I and II. These operations can examine or change any individual binary bit within a character in any position of memory. Two operations, transfer on zero bit and set bit, expand the control of data by the program and further conserve memory space. Bit manipulation also provides a rapid, simple method of program switching. By this means, a number of switches can be built into a single character; the presence of a 1 or 0 determining the conditions of on or off. Any practical number of switches can be set or tested without using an ASU or placing a constant in memory.

Set Bit (% — SB) *(705 III)*

The set bit instruction sets any bit in any character to either 0 or 1. The address of the instruction specifies the location of the memory character (Figure 147). Zoning over the tens and hundreds positions of the address specifies the particular bit within the character to be set. The mnemonic and numerical codes are written on the program sheet as follows:

SB	01-06	(SBZ)	Set bit zero.
SB	07	(SBA)	Set bit alternate.
SB	08	(SBR)	Set bit redundant.
SB	09-14	(SBN)	Set bit to 1.
	01		Set 1 bit to 0.
	02		Set 2 bit to 0.
SBZ	03		Set 4 bit to 0.
	04		Set 8 bit to 0.
	05		Set A bit to 0.
	06		Set B bit to 0.
SBA	07		Reverse A bit.
SBR	08		Reverse C bit.
	09		Set 1 bit to 1.
	10		Set 2 bit to 1.
SBN	11		Set 4 bit to 1.
	12		Set 8 bit to 1.
	13		Set A bit to 1.
	14		Set B bit to 1.

An SB instruction with a numerical coding of 08 causes the check bit to be reversed, regardless of its condition. This instruction may be used to develop an invalid character in memory for test purposes.

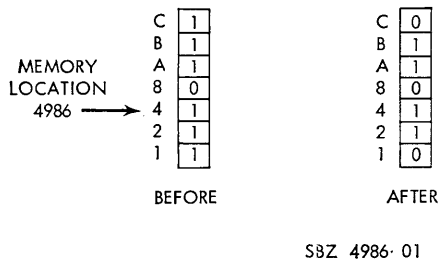


Figure 147. Set-Bit Operation

In all other cases, changing the zone or numerical bits of a character automatically inserts or deletes the check bit to place a valid character in memory.

Transfer on Zero Bit (. — TZB) *(705 III)*

This instruction is preceded by a receive instruction setting MAC II at the location of the character in memory to be tested (Figure 148). The numerical portion of the TZB address is the location of the next instruction to be executed if the bit tested is 0. Zone coding, similar to ASU coding, is placed over the tens and hundreds positions of the TZB address to specify which bit in the character is to be tested. This coding is as follows:

BIT	ZONING
1	01
2	02
4	03
8	04
A	05
B	06
C	07

If the bit tested is 0, a transfer is made.

The TZB and SB may be used for the purpose of either recording or recognizing a number of variables placed in one or more characters. For example, data that can be expressed in such terms as yes or no, on or off, up or down, can be indicated by 1 or 0. For example, the presence of a 1 in a specified position of a character may indicate that a deduction is to be taken from gross pay. A 0 indicates that no deduction is to be taken. A 1 may indicate in personnel records that the individual is a naturalized citizen; 0, that he is an alien. A 1 might indicate male; 0, female.

In setting up code characters within a record for the purpose of recording variables, several limitations must be considered.

1. The variables must not be recorded in such a way that a character can be formed of all zeros. This configuration is a storage mark and cannot normally be contained in memory.

2. The characters must not be all 1's. This configuration constitutes a group mark and may improperly limit a writing operation.

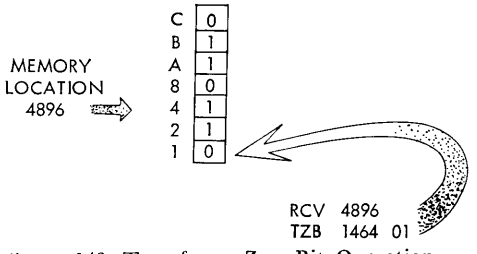


Figure 148. Transfer-on-Zero-Bit Operation

3. If the code character is to be printed, the programmer must insure that every possible combination of bits forms a character that can be written by the printer.

Record Arrangement for Printing

When a write instruction follows the selection of a printer, information is transmitted from memory from left to right, beginning with the character specified by the address part of the write instruction. Information is sent to printer record storage exactly as it was stored in memory. Therefore, the arrangement of the record in memory must be programmed to conform to the report form before printing occurs.

Each position in the record uses one print wheel or printing position in the printed line. Insignificant zeros to the left of digits in arithmetic fields are normally changed to blanks before printing by use of the store-for-print instruction. Indicative fields, descriptions, or other portions of the record can be shifted in memory to conform to the printing arrangement by load and unload, receive and transmit, or send instructions.

Store for Print (5 — SPR)

The store-for-print instruction normally is used to transfer a numerical field from the accumulator or auxiliary storage to memory. The address of the instruction specifies the location in memory where the field is to be stored and the storage unit to be used (Figure 149).

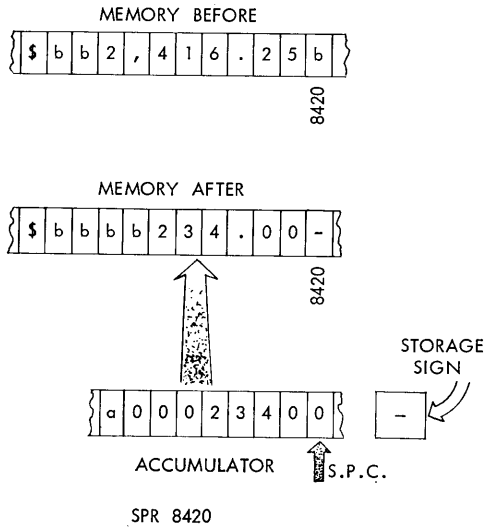


Figure 149. Store-for-Print Operation

The sign of storage is placed in the address location of memory as a blank character, if the sign is plus, or as a dash if the sign is minus. The numerical field is then stored to the left of the sign position.

When the store-for-print instruction is used, the area in memory that receives the field must contain commas and/or a decimal point in positions that properly point off the digits as they are to be printed. When these periods or commas are encountered in memory during the store operation, they are skipped. However, insignificant zeros, characters with zero numerical portions, and commas to the left of these insignificant positions are replaced in memory with blanks.

The field in storage is not affected by the instruction.

Normalizing Accumulator and Auxiliary Storage

The normalize-and-transfer instruction is useful in removing zeros, one at a time, from the left end of a factor in accumulator or auxiliary storage. A program routine may then be inserted to count the number of zeros removed, a necessary function in printing a floating dollar sign or in filling in blank spaces to the left of a field with asterisks for check protection.

Normalize and Transfer (X — NTR)

The normalize-and-transfer instruction removes the left-hand character of the storage field if the numerical part of that character is a zero.

A transfer is made to the location specified by the address part of the instruction when the zero is deleted. If the numerical part of the left-hand character is not a zero, the storage field is not changed, a transfer is not made, and the machine proceeds to the next instruction. If the field is a single zero, it is not deleted and a transfer is not made.

Control Instructions

Six control instructions control the various features of the input-output units and turn on and off the input-output indicators. A seventh control instruction, available only on the 705 III, is used to skip a section of defective tape. A control instruction always applies to

the last selected unit. The address part of the control instruction specifies the feature to be controlled.

Control 0000 (3 — IOF)

The input-output indicator of the last selected unit, if on, is turned off. The instruction refers to printers, tape units, drums and card readers.

Control 0001 (3 — WTM)

A tape mark is written on tape by the last selected unit. The writing of this single-character record is checked in the same way as the writing of all characters from memory.

Control 0002 (3 — RWD)

The tape on the last selected unit is rewound.

Control 0003 (3 — ION)

The input-output indicator on the tape unit last selected, if off, is turned on. The instruction is used with tape units only.

Control 0004 (3 — BSP)

The tape on the last selected unit is backspaced to the previous inter-record gap.

Control 0005 (3 — SUP)

This instruction applies to printers and punches only. It prevents printing or punching of information from record storage for one cycle. The instruction is normally used to prevent printing or punching when a read-write error has occurred from memory to record storage. Under program control, the record can be reloaded from memory after the error condition has been recognized.

Control 0009 (3 — SKP) *(705 III)*

The tape on the last selected unit is skipped forward approximately five or six inches. During the skip, the tape is erased as it passes over the read-write head.

The instruction is intended for use with the 729 tape unit with a two-gap head. When a writing error persists after two or three attempts, the tape may be backspaced once more and skipped over what may be presumed to be an imperfection in the tape itself.

Transfer Instruction

Transfer on Signal (O — TRS)

The transfer on signal causes a program transfer when the last selected indicator is on. The indicator may be an input-output indicator, a check indicator, or an alteration switch. The transfer is made to the location of an instruction specified by the address of the TRS.

When the transfer on signal is executed, the selected check indicator is automatically turned off.

The instruction has two modes of operation when used on the Models I and II. The first is with no zone coding over the tens and hundreds positions of the address. This causes a transfer as described above. The second mode is with 01 zoning, in which case the instruction becomes a transfer-on-ready for use with the IBM 777 Tape Record Coordinator.

In the 705 III, the TRS with no zone coding serves the same purpose as when used with the 705 I or II. Its operation is dependent upon the condition of the previously selected indicator.

The second mode of operation with the 705 III involves expanded use of zone coding. The instruction can then test the various indicators without the use of a previous select instruction. Codes and corresponding mnemonic abbreviations are listed below.

ASU		MNEMONIC	INDICATOR
01	(TRR)	Transfer Ready	Ready; DS or TRC
02	(TTC)	Transfer Transmission Check	PCT Data Check
03	(TSA)	Transfer Synchronizer Any	PCT or I-O (Tape)
10	(TIC)	Transfer Instruction Check	Instruction Check (0900)
11	(TMC)	Transfer Machine Check	Machine Check (0901)
12	(TRC)	Transfer Read-Write Check	Read-Write Check (0902)
13	(TEC)	Transfer Echo Check	Record Check (0903)
14	(TOC)	Transfer Overflow Check	Overflow Check (0904)
15	(TSC)	Transfer Sign Check	Sign Check (0905)

Alteration Switches

Six alteration switches are provided on the operator's console with addresses of 0911 through 0916. The operator may turn them on or off manually.

Switches are selected in the program in the same way as any other component in the 705 system. Any switch can be specified by the address part of a select instruction (SEL 0911).

On the 705 III a transfer-any instruction may also test the condition of an alteration switch in the program. When the switch is on, a transfer is made to

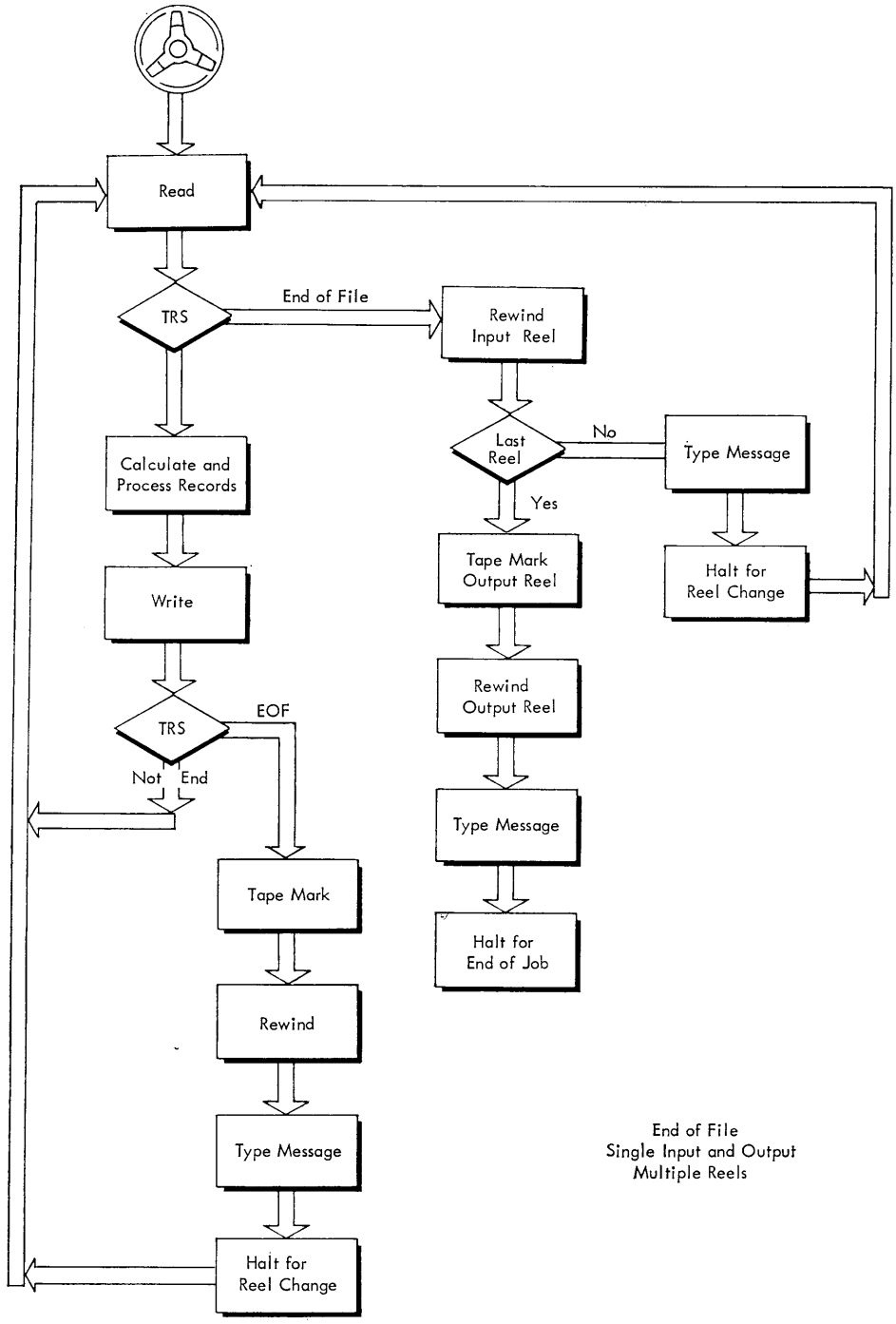


Figure 150. End of File, Flow Chart

the memory location specified by the address part of the transfer-any instruction. When the switch is off, no transfer is made and the 705 proceeds with the next sequential instruction.

End-of-File

Each input or output unit in the 705 system, except the card punch and typewriter, is equipped with an

indicator to signal an end-of-file condition. Whenever a unit is selected by a program instruction, the input-output indicator associated with that unit is also automatically selected.

An indicator is either on or off. Once an indicator is turned on, it remains on until it is turned off either by the program or by a manual operation. The status of an indicator is tested or interrogated by a transfer-on-signal instruction in the program. The instruction usually follows immediately after reading or writing operations. If the indicator is on, a transfer is effected to the memory location specified by the address of the instruction. End-of-file instructions are normally included as subroutines in the program. When the indicator is off, a transfer-on-signal has no effect and the machine continues to the next instruction.

An end-of-file or branch routine may be arranged in a variety of ways depending upon operating conditions. For example, the typewriter may be used to notify the operator that a tape unit is in end-of-file condition. The machine may then be programmed to stop while reels are changed or to select automatically an alternate unit and continue operation. Other control instructions may automatically rewind a completed reel.

By pre-arranged alteration switch settings, the operator can, after putting in the last group of cards, make the machine automatically select program instructions to continue operation after an end of file has been signaled. Such operation might include final total calculation and printing.

When two or more tape units are used to read or write a single file on multiple reels, an end-of-file signal on the first reel can change the select instruction address to specify the second tape unit. Reading or writing can then continue on the following reel without loss of operating time while the preceding reel is rewound. Reading can continue alternately between two units until the mounting of the last reel is noted by an alteration switch setting.

When several related records are processed in or out of the system during the same procedure, an end of one input record file can cause program modification to consider only those records remaining to be processed. Calculation or processing steps for the records on the completely processed file are then ignored.

Figure 150 is a flow chart of an end-of-file procedure for single input and output tapes.

Tape Unit Indicator

Each tape unit is provided with an input-output indicator to indicate the end of a reel or file. The indicator can be turned on by any of the following:

1. Sensing the end-of-reel marker while writing.
2. Sensing the tape mark as a unit record while reading.
3. Using an ION instruction in the program when a tape unit was last selected.

The indicator can be turned off by:

1. Depressing the unload key on the tape unit to remove the reel.
2. An IOF instruction in the program when a tape unit was last selected.

Card Reader Indicator

Each reader is provided with an input-output indicator that is turned on when a read instruction involving that reader is given after the last card has been read from record storage. The indicator is turned off by:

1. Loading record storage by feeding cards.
2. An IOF instruction when the reader was last selected.

Printer Indicator

Each printer is provided with an input-output indicator to indicate the end of a printed page. It is turned on by the overflow signal obtained from channel 12 of the carriage tape. The indicator is turned on after the next write instruction involving that printer is given (Figure 151). The indicator is turned off by:

1. Using an IOF instruction in the program when a printer was last selected.
2. Depressing the printer start key.

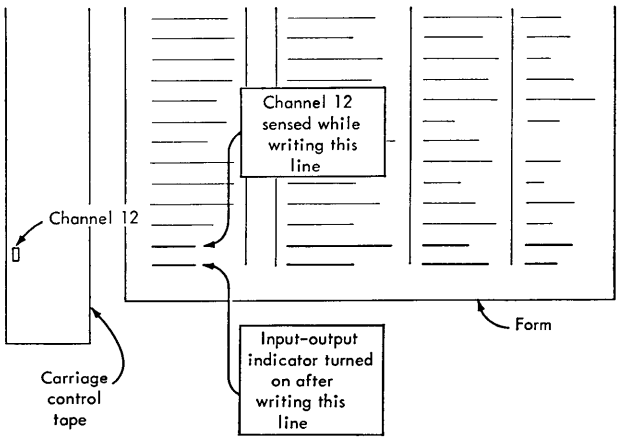


Figure 151. Printer, Input-Output Indicator

Drum Indicator

The input-output indicator of a drum is turned on if an attempt is made to read or write beyond the limits of the drum. The indicator is turned off by using an IOF instruction in the program when a drum section was last selected.

Transfer Any (I — TRA)

The test for an end-of-file condition or the condition of a check indicator may be simplified in the main program by using a transfer-any instruction.

The transfer-any indicator is turned on whenever an input-output or check indicator is turned on. When the indicator is on, a transfer is made to the memory locations specified by the address part of the instruction. The transfer-any indicator is turned off by the transfer itself.

The TRA instruction address without zone coding has a single mode of operation for the 705 I and II. However, with the use of zone coding on the 705 III, the TRA may also be used to interrogate an alteration switch as follows:

ZONE CODE	ALTERATION SWITCH	MNEMONIC
01	0911	TAA
02	0912	TAB
03	0913	TAC
04	0914	TAD
05	0915	TAE
06	0916	TAF

Use of a TRA with the coding listed above performs the dual function of selecting and testing a specific alteration switch and transferring if the switch is on.

All procedures perform two functions. First, they accomplish useful work; second, they control such factors as the amount, quality, and accuracy of the work. A data processing procedure is no exception.

Here, the useful work consists of such operations as sorting, calculating, collating, reading, and printing. Control operations are necessary to establish and maintain accounting controls, record counts, calculation checks, and machine checks. Controls or checks may be broadly classified under three headings: machine checks, system checks, and program checks.

Machine Checks

The 705 makes six specific checks upon data being processed. Each type of check is associated with a neon indicator and a manual switch on the operator's console. When a neon is on, an error condition is indicated.

The switch associated with each neon indicator can be manually set to either AUTOMATIC or PROGRAM. When set to AUTOMATIC, an error detected by the corresponding check indicator causes the machine to stop. When the switch is set to PROGRAM, the corresponding indicator may be interrogated by a program instruction to turn off the indicator and start corrective action automatically.

In many cases, therefore, it is not necessary to interrupt machine operation when an error condition is detected. The program can include branch or subroutines 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 machine operation continues. If the error persists, machine operation can be interrupted or the incorrect record can be noted and operation continued.

A check indicator is interrogated by two instructions: select, followed by transfer on signal. The select instruction specifies the proper indicator. The transfer-on-signal address transfers the program to the first instruction of a subroutine which is to be followed if an error is detected. The transfer is made only when the indicator has been turned on by an error condition. Machine operation is not interrupted when the error is corrected by the branch program. The transfer-on-signal instruction turns the indicator off.

Check indicators and their assigned addresses are:

Instruction Check Indicator	0900
Machine Check Indicator	0901
Read-Write Check Indicator	0902
Record Check Indicator	0903
Overflow Check Indicator	0904
Sign Check Indicator	0905

Instruction Check Indicator 0900

The instruction check indicator turns on when the following conditions occur:

1. A character code error is detected during instruction time.
2. An invalid operation part is encountered in the operation register.
3. The operation part is incorrectly interpreted.
4. The units position of the address part of any transfer instruction, or a transmit instruction specifying accumulator 00, is not 4 or 9. The send instruction is also checked on the 705 III.
5. The field addressed by an indirect address coded instruction is not a position ending in 4 or 9. It is recommended that the switch associated with this indicator be turned to AUTOMATIC to cause a machine stop when an error is detected. Programming around this type of error is usually impractical. With the switch set to AUTOMATIC, the machine stops during the character cycle in which the error occurred.

Machine Check Indicator 0901

The machine check indicator is turned on when a character code error is detected during the execution of all instructions in which data are transferred from accumulator or auxiliary storage or memory.

When the indicator switch is turned to AUTOMATIC, the machine stops during the character cycle in which the error occurred except if an error occurs during the execution of write or write and erase. In this case, the indicator is turned on but no automatic stop occurs. Such an error may be detected by programming, as the read-write check indicator is also turned on.

Read-Write Check Indicator 0902

The read-write indicator turns on when a character code error is detected during the execution of a read,

write, read-while-writing, or write-and-erase instruction. The indicator also turns on when an error is detected in reading the holes in the card or by the longitudinal check in tape reading. The indicator, therefore, checks the transmission of data from all input units to memory. It also checks the transmission of all output data from memory to the drum, tape unit, card punch record storage, printer record storage, and typewriter. The indicator turns on if an attempt is made to read or write beyond the limits of the drum or if an error occurs in recording a tape mark.

When the indicator switch is turned to AUTOMATIC, an error stops the machine after the instruction is executed.

Record Check Indicator 0903

The record check indicator turns on when an error is detected by the brush-compare method on the punch and by the echo-check method on the printer. An error in card punching is detected as the card passes a brush station after it has been punched. If an error occurs, the record check indicator turns on during the execution of the second write or write-and-erase instruction after the instruction that punched the error card.

An error in printing is detected by sensing the position of each print wheel during the print cycle. If an error occurs, the indicator turns on during the execution of the next write or write-and-erase instruction involving that printer.

In both cases, when the switch for this indicator is on AUTOMATIC, an error stops the machine at the end of the punching or printing cycle during which the indicator was turned on. At this time, the error card is the last card to go into the punch stacker. The incorrect line of printing immediately precedes the last printed line.

Overflow Check Indicator 0904

The overflow check indicator is turned on during an add or subtract operation when the number of digits in the result is greater than the number of digits in the longer of the two fields. An overflow is indicated as a result of a round operation, if a carry-over is made out of the high-order position of the accumulator storage field.

The indicator is turned on by a divide instruction when the divisor does not have a greater absolute value than an equal number of digits taken from the left end of the dividend. When the error switch for this indicator is turned to AUTOMATIC, an error stops the machine during the execution of the instruction.

Sign Check Indicator 0905

The sign check indicator turns on if a field addressed by a reset and add, add, reset and subtract, subtract, multiply, or divide instruction does not have plus or minus zoning over the right-hand digit.

When the switch for this indicator is on AUTOMATIC, an error stops the 705 I or II during the character cycle following the one in which the error was detected. For 705 III operation, the sign check occurs and stops the machine in the same cycle in which the error is detected.

System Checks

System checks can be defined as any checks other than those made by the built-in check circuits in the 705. This is a broad category and includes such program checks as record counts, hash totals, control totals, proof figures, limit checks, and crossfooting balance checks. All of these checks can be programmed and are useful tools for program checking.

The system checks to be incorporated in a program should be designed during the original planning phase. What kind of system check to use depends upon the program to be checked. System checks designed for a specific program generally are unique to that program. Some general techniques applicable to any program are described here.

Magnetic Label

File identification placed at the beginning of a reel of tape is referred to as a magnetic label. The label may specify the job title 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 reel to designate the end of the file. The labels are read into memory at the beginning and end of the program as an added control that the proper records have been processed. The label may also assure a true end of file or end of job.

A counter may be included as part of the label to record the number of passes to which the tape has been subjected. Old tape may consequently be retired from active files before excessive wear has occurred.

Record Count

A record count is simply a count of the number of records in a file. This count is made each time the file is written and is carried as an additional record at the end of the file. The count is made again when

the file is being read for processing to see that all records in the file have been read in.

Hash Total

A hash total is a total of an important numerical or alphamerical field (such as part number) for all records. It checks that all of the records written on the last processing run have been read in during the present cycle. It is similar to a record count, except that the hash total gives an additional check that all part numbers have been read in correctly. The hash total is carried as an additional record at the end of the file. This total may be computed as the original tape is written, or during a subsequent machine run.

The hash total may also be computed for certain vital fields in a single record. This total is carried as an additional field in each record and can be checked whenever that record is read into memory. Hash totals must be accumulated by use of add-to-memory instructions if alphamerical fields are part of the total.

Control Total

A control total is a predetermined total of some amount or quantity field in a file of records. During the processing, a sum of this field is accumulated and checked against the count total. The control total can be in the form of a grand total for all input data, or an intermediate or minor total for each control group in the file. An example of the use of control totals is a simple payroll where a predetermined total is made of the employee hours per pay period. During the processing of the payroll, a total of hours per employee is accumulated and, at the end of the program, the two totals are compared.

Proof Figures

Proof figures are sometimes used to check an important multiplication in a program. The proof figure is usually additional information carried in the record. An example of this is the multiplication of quantity by cost required in grocery billing. The check is based on a relationship between cost and a so-called proof cost. An arbitrary fixed figure *Z*, larger than any normal cost, is set up. Then the proof cost is expressed by the formula: $Cost + Proof\ Cost = Z$.

When quantity is multiplied by cost, it is also multiplied by proof cost. Normally, two of the totals needed for the check, quantity and quantity times cost, are accumulated during the program. The other factor needed for the check (quantity times proof cost) is also accumulated in the program. Now it is possible at any point to check as follows:

$$(Quantity \times Cost) + (Quantity \times Proof\ Cost) = (Quantity \times Z)$$

The left side of the equation can be calculated by a single addition of the two progressive totals accumulated during the program. The right side of the equation can be calculated by a multiplication of the accumulated quantity and the factor *Z*. This check insures that each particular multiplication was performed correctly. This type of check applies to other applications by the same general approach, that of adding check information.

Limit Check

A limit check is the test of a field in a record or a total in the program to see if certain predetermined limits have been exceeded. An example of this would be a transaction code known to include only numbers 0 through 5. In the program, a check should be made to see that the code does not exceed 5.

Another limit check applies to reasonableness. For example, certain totals are known to vary not more than ten percent between processing cycles. This check can be easily programmed.

A further use of this check is in a table look-up operation. If a value is known to be in a given table, the modified table address may be checked against the address of the upper table value to verify correctness of the search. If the file search begins to exceed the limits of the table, an error has occurred and corrective action should be taken.

Crossfooting Balance Checks

Crossfooting balance checks are useful in many programs. An example is in payroll calculation. During the processing of each record in a payroll, independent totals are accumulated of gross pay, taxes, miscellaneous deductions, and net pay. These totals can be crossfooted and checked at any point in the program. For example, the total gross pay at any point should equal total net pay, plus total deductions, and taxes.

Program Checks

Program checks are normally designed to examine and verify the execution of certain routines or particular instructions in the program that may not be satisfactorily covered by system or machine checks. Like

system checks, they are accomplished by programming. Many types of program checks may be used. They include checks of arithmetic operations when amounts or quantities are being originated in the procedure and no previous control total exists. For example, the payroll calculation of gross pay for each individual employee cannot be verified by control totals.

A check can be made of comparisons or upon the sequence in which instructions are executed. Checks can be devised to check recognition of plus or minus balances, reading or writing into proper locations, selection of proper drum sections or other units, backspacing, and others.

The programmer should always first use the readily available checks, including machine and system checks. However, if these checks do not satisfy the requirements of procedure quality control, the characteristics of the problem itself may be considered for additional checks. A problem often contains self-checking or limiting factors that may furnish suitable systems or program checks. When necessary, the problem should be modified during early stages of planning to include program checks.

Check-Point Procedure

A check-point procedure is a programmed checking routine performed at specific processing intervals or check-points. Its purpose is to check that the program has been performed correctly to a predetermined point. If it has, then the status of the machine is written out or "remembered" periodically. The program is then continued until the next check-point is reached.

If an error is detected at a check-point, or during the processing interval between check-points, the procedure is backed up to the previous check-point. The machine is then restored to its exact status as recorded on the check-point tape or on the drum. The entire program between check-points is rerun.

The operation of "backing up" requires that the record of machine status, as originally written on the check-point tape or on the drum, be restored in memory, Also, it requires that all input and output tape units must be returned to their positions at the time the last check-point was recorded.

However, check-point can also be used in procedures involving printing or card punching. When restarting, proper identification of the output group of cards or printed records containing the error must be made so that these records can be removed at the end of the job. When using the card reader, manual intervention is necessary to restart.

Restart Procedure

A restart procedure is a programmed routine designed to return the machine automatically to some predetermined point in the problem. This is normally the preceding check-point.

The restart accomplishes two things:

1. It "backs up" the entire machine system to the predetermined point in the problem. Tape units are back-spaced automatically; card units and printer are adjusted manually.
2. It restores the memory of the machine to its status at preceding check-point. Only certain selected portions of memory may be restored if desired.

The purpose of the restart procedure is to reduce the need for manual intervention in case of error. The routine may be executed automatically when an error is discovered at check-point or it may be entered from any point in the program when an error condition is recognized.

If a stop occurs while a program is running, one or more of the following factors will have caused it:

1. Data errors.
2. Operator errors.
3. Random machine errors.
4. Emergency repairs or power failure.
5. End of shift or job interruption.

The operator has a choice of procedures to put the machine back in operation. He may make the correction, rerun all or a portion of the program, adjust the machine (maintenance), or shut the system down. It is a great advantage to include programming to correct automatically steps 2, 3, and 5 above.

The program may instruct the machine to "try again" in the case of a random machine error or to check the problem at regular intervals so that, if a rerun is necessary, only a portion of the program need be rerun.

The first approach to a restart is to establish some point in the program to which the program will return in the event of a stop. The simplest procedure is to merely return to the original starting point of the job. The program includes a transfer to "housekeeping" instructions to rewind all tapes, turn off input-output indicators, set switches, check tape reel labels, and so on. In such a restart procedure, no record of memory is needed.

A second approach is to establish a break or recovery point at the end of a reel. Checking for errors, storing of accumulated totals, checking reel labels or recording of other pertinent information is done before or after each reel change. The end of a reel then signals a break point and all other reels are changed

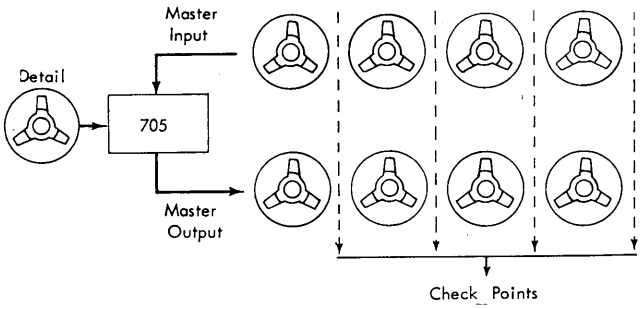


Figure 152. Schematic, Recovery Point at End of Reel

at the same time, whether at end of file or not. If a restart is necessary, the system recovers to the beginning of all files.

This second method, however, is practical only when both input and output reels are of approximately equal length. The effect is to divide a multi-reel application into a number of single-reel jobs. Figure 152 shows this method. In this case, assume that files consist of four reels of master input, four reels of master output, and one reel of detail transactions.

Detail records are merged with masters and do not add records to the master input file. Recovery points are established at the end of each master input reel, dividing the application into four smaller, independent runs. Control totals and other restart information are written on the output tape at the beginning of each master reel as a special record. This record might be an addition to the reel label.

If a restart is necessary, the program rewinds the master input, output, and the detail reels. The totals are reestablished and the detail reel is then read forward by the restart routine until a record matching the first master on the input reel is located. The job is then continued in the normal manner. Maximum time lost by the restarting procedure is the time required to process and rewind one full reel of tape. If a power failure or other interruption occurs, the same procedure can be used, except that the program must also be reloaded.

General Consideration of Check-Point and Restart

The problems of an application must be viewed when planning for restart procedures.

1. Checks may be taken, before printing, of totals when group control occurs. These checks include 0901, 0904, and 0905. In the event of an error, a transfer to a restart procedure is made to return the entire machine to the last error-free check point. A sign check usually indicates an error in input data. Therefore, a programmed stop when this error is detected may be advisable. Indicator 0900 is normally set to AUTOMATIC. The operator then has an option to make corrections when an instruction error occurs or to transfer to a restart without continuing to the next check point. The program may be reloaded or read in automatically as a part of the restart procedure.

2. System checks may be included at check point. Restart may not always be desirable, if a systems error is detected. For example, if a control total is out of balance and the record count shows that a record is missing from an input file, a rerun of this section of the file cannot locate the missing record. In this case, the check-point should indicate the cause of error and stop the machine with operator option either to continue or to take corrective action.

3. Normally, the check-point routine is modified depending upon the point at which it is taken. For example, when system checks are also taken at check-point, a more complete check occurs than if check-point is forced at periodic intervals.

4. Read-write errors are not tested at check-points. They should be reprocessed as they occur and the operation stopped if they persist.

5. Jobs running 15 minutes or over should normally include check-points with automatic restart. Jobs without check-point might extend for longer periods of time if they are to be run infrequently. Validity of results must be checked for any job, regardless of its length. However, for jobs of less than 15 minutes' running time, it is generally more economical merely to rerun the job in case of error (not read-write errors), rather than to devote additional programming time to comprehensive checking.

Indirect Addressing

All instructions illustrated in the preceding sections have been shown using direct addresses. That is, the address of an instruction refers directly to the location of data or other instructions in memory, or to the address of a machine component. Control and shift instructions use the address to specify the type of control to be exercised or the number of positions of a factor to be adjusted.

Any instruction executed by the 705 III, however, can use an indirect address. Such an address does not refer directly to the data to be processed nor to a machine component. It refers to a memory location which *contains the address* of the data, device, or control function.

Such a form of addressing can be particularly useful in performing the operations of address modification. In a program, it may be necessary to address a number of instructions to a single machine unit, such as a tape unit. The unit may be selected for reading, for error routines, in restart procedure, or in various other branch routines. If the unit is to be alternated with some other unit, all the addresses of all instructions involving that unit must be modified. A considerable number of other instructions might be needed for this purpose.

However, if the select instructions involving the specified tape unit were indirectly addressed to *one* memory location, that location could contain *one* address of the tape unit. Therefore, to modify all select instruction addresses, it would be necessary only to modify the single address to which the select instructions refer (Figure 153).

In memory, an indirect address is identified by a 1 in the A-bit position of the units position. Any address may be indirect, but the numerical portion of the units position must be either 4 or 9. In the text, an indirect address is indicated by an asterisk.

Any number of indirect addresses throughout a program may refer to a single effective address. All ASU zoning and the A bit position of the low-order character of the effective address are ignored. For example, the following instructions add some constant (located at 1004) into ASU 01, and then subtract the same constant from ASU's 02, 03, and 04.

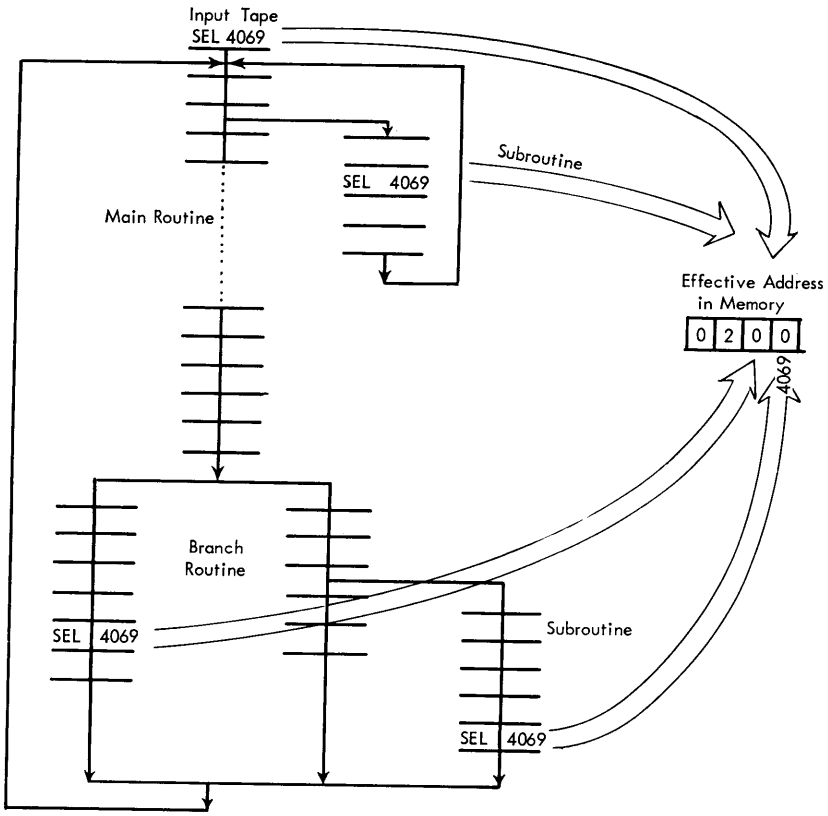


Figure 153. Indirect Addresses

0104	ADD	01	1004
0109	SUB	02	0104*
0114	SUB	03	0104*
0119	SUB	04	0104*

The address at 0104 carries ASU coding which is ignored by the indirect addresses at 0109, 0114, and 0119.

Relationship between Computer and Programmer

It is a rather ironic fact that present-day computing equipment, primarily designed to save work, has, at the same time, produced a new kind of drudgery. This is the task of program writing. And, since the program is the only means of communication between man and machine, it is necessary that some method be devised to make this communication as simple and as effective as possible.

The problem is mainly one of language. While it is not impossible to write intructions in the machine language, there are many reasons why such an approach is inefficient. First, in the 705 as in other systems, the programmer must memorize the actual operation code. Second, he must calculate the exact location of all instructions and data in order to use actual addresses in his instructions. The problem of converting these addresses to the proper characters (that result from zoning the high-order addresses of memory, from ASU zoning, and from indirect addresses) becomes so complex that such a system is entirely impractical. Even if this could be done, one clerical error in establishing a location of either data or instructions would make the program useless. One omission of an instruction would mean the entire relocation of all information following an inserted instruction. It should be apparent that program writing in actual machine language is only effective for short programs consisting perhaps of a few dozen instructions at most.

Figure 154 shows the basic relationship between the computer and programmer. First, the problem is analyzed and reduced in terms of the operations that the machine can perform. The programmer may use tables, formulas, codes or other aids, applicable to the specific situation. When the program is written, it is solved on the computer. The problem then becomes input data and the machine, by calculation or other means, produces useful output.

Various systems have been devised to make machine and human language more compatible. Generally, the approach is one of inserting an intermediate process between the statement of the problem and the production of a final program. To do this, the programmer states the problem in a language more nearly like his own, but in definitely restricted or standard terminology.

For example, he may be allowed to write the program in abbreviated or mnemonic fashion, adhering to pre-established rules and restrictions. By doing this, he is able to call his data by the appropriate name, such as input record, gross pay, quantity, and so on,

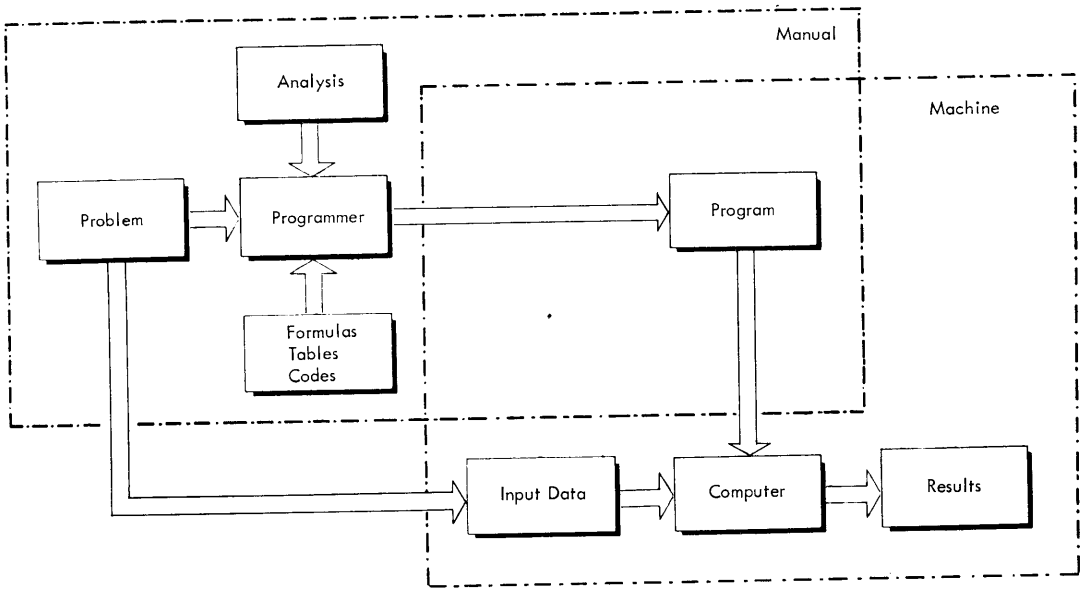


Figure 154. Direct Conversion of Problem to Machine Language

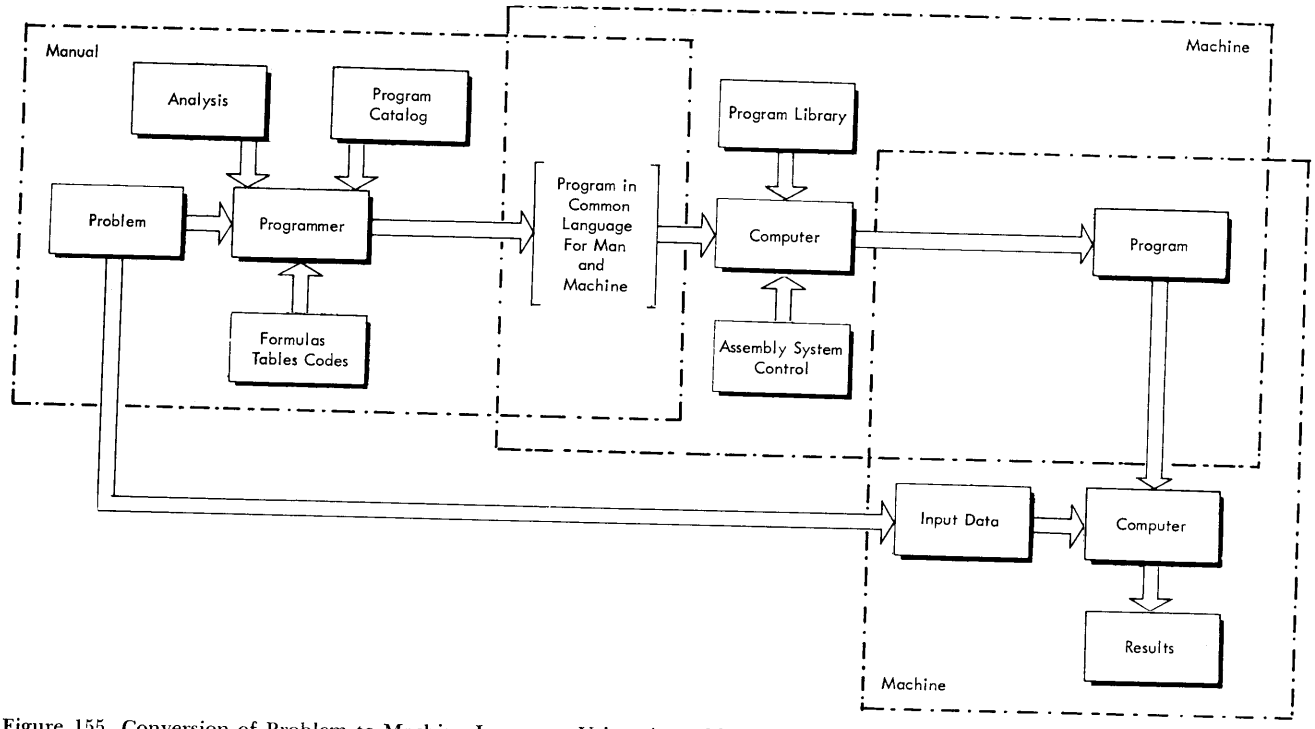


Figure 155. Conversion of Problem to Machine Language Using Assembly System

rather than by an address in memory. He is also able to allow the necessary amount of space for data but does not need to calculate the actual locations of either instructions or data. The system may also provide other features enabling him to draw upon previously written and tested routines usable for sections of the program he is writing. It may also incorporate operations that are not actually in the machine's vocabulary but which are converted to these terms by the intermediate process.

Figure 155 shows the basic flow of the work between problem and output, using a program system. Again, the programmer analyzes the problem, writing a program in pseudo-language according to set rules. He can now draw on other available programs for some parts of his program, assuming that this material is in the "library" of the system for his use. This program then becomes input data to the intermediate process, called an "assembly." The program is punched in cards for direct input, although for long programs, a card-to-tape operation may be performed first.

The machine now takes the punched program data and converts it to an actual program in its own language. The processing is under control of the assembly system. A machine program is the only output of this process, although supplementary listing or tape may usually be obtained.

The resulting program is now stored in the computer and the problem becomes input data as before. The operation is now one of converting the problem input to useful output.

Autocoder

The Autocoder is a system of program writing, developed and tested by IBM for the 705. It offers a number of important advantages to the programmer:

1. It saves clerical work. The programmer merely indicates the amount of space in memory required for data. The assembly locates all instructions automatically and reserves other areas as indicated.
2. The programmer can use all 705 operations plus an expanded set of autocoder operations.
3. The assembly process edits the program for clerical errors, such as items out of sequence, references to wrong data or instructions, and incorrect operation codes.
4. The system makes available to the programmer a library of subroutines to use in his own program, as required.

IBM AUTOCODER PROGRAM SHEET

705 DATA-PROCESSING SYSTEM

CODED BY

CHECKED BY

PROGRAM STOCK STATUS IDENT. A-1460 Inserts on back

DATE

PG	LINE	TAG	OPERATION	NUM.	OPERAND	COMMENTS
01	01	START	SEL		200	
	02		RD		MASTER INP	
	03		SET		18	
	04		RAD		2ADDRESS	Get transfer address
	05		UNL		2TRANSFER	Reset transfer address
	06		SEL		202	
	07		RD		DETAIL	
	08		LOD		IDET CODE	
	09		CMP		IMAS CODE	
	10		TRE		EQUAL	

Figure 156. Example, Autocoder Program

Autocoder Organization

As in any system so far developed, all information for a 705 program is originally supplied by the programmer. This information is written on a program sheet (Figure 156). Machine instructions, record areas, work areas and all other data are described in complete detail. Each separate item of information forming a program entry is written on one line of the sheet.

Entries are written in the sequence in which they are to be stored in memory, although some other sequence may be specified. A space is therefore provided for writing program page numbers. Line numbers are preprinted on the sheet. Entries may include both instructions and data.

After the program has been written and checked, it is key punched into autocoder instruction cards (Figure 157). Each card contains one program entry (one line from the program sheet). Information is punched exactly as written including page and line number. An identification for the entire program may be gang punched into all cards.

Punching is next verified for accuracy by the card verifier or by manually checking the original sheet with listed or interpreted cards. After verification, the cards are sorted to line number, then to page number. Inserted entries, if any, are placed in proper sequence at this time.

The punched instruction cards may then be used as direct input to the 705, or cards may first be converted to tape to provide a faster input to the assembly. In either case, the program instructions are one input to the 705. The autocoder system tape is the second input (Figure 158).

The autocoder system tape contains several types of information:

1. System control: instructions to load the autocoder into the 705 memory and miscellaneous instructions to control the form of input and output as directed by the console operator.
2. Librarian: instructions to control the revision to the library.

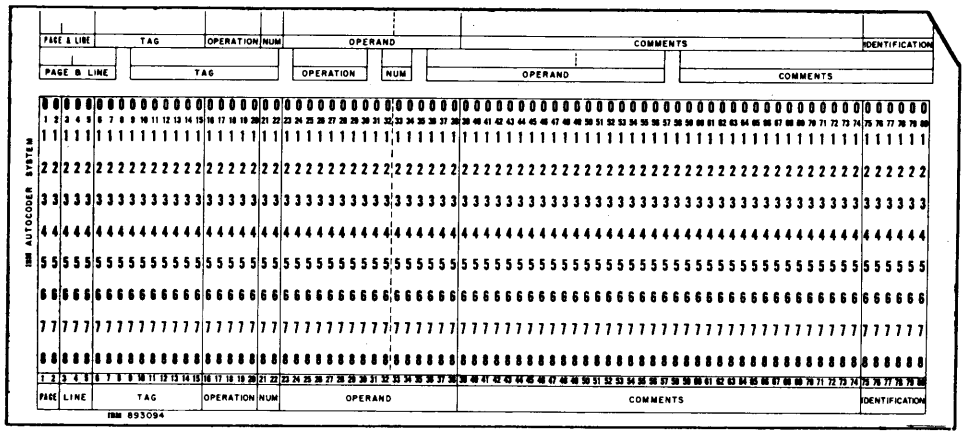


Figure 157. Instruction Card, Autocoder

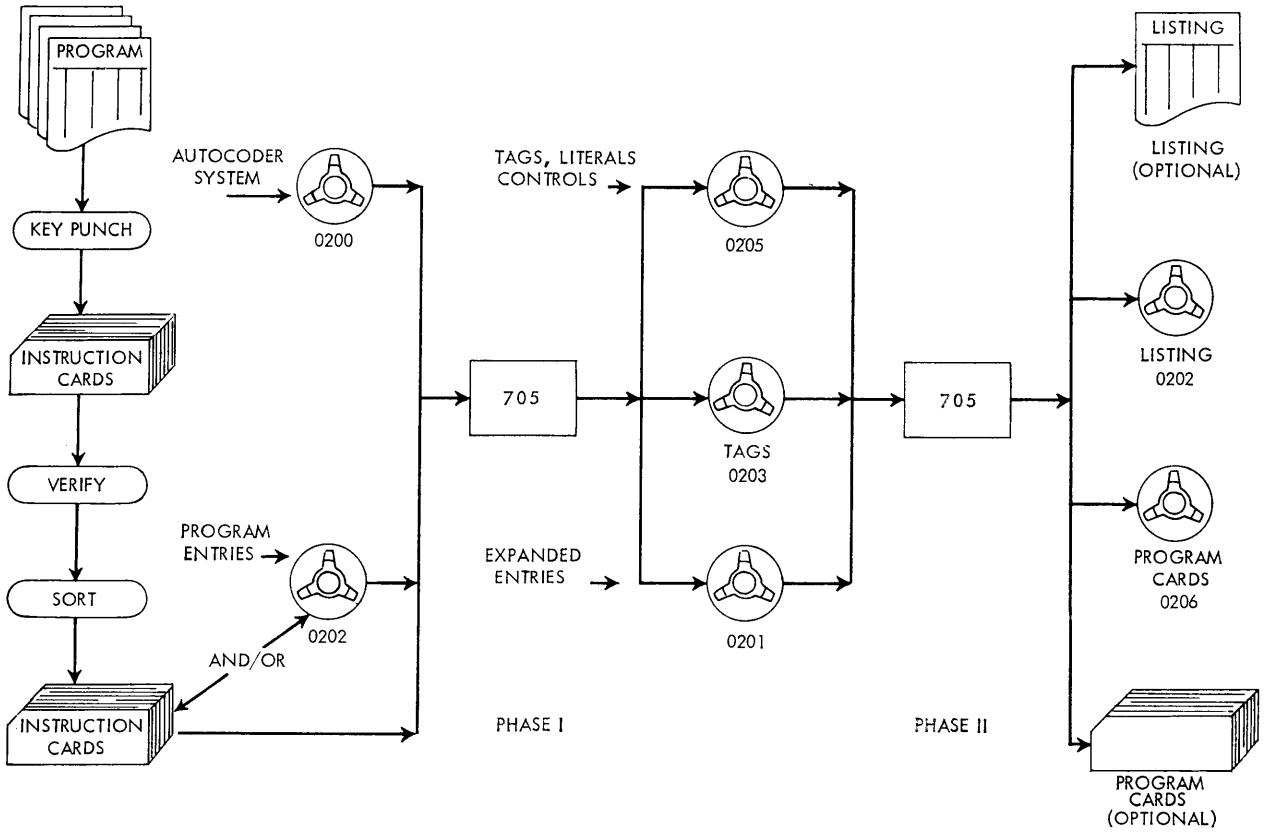


Figure 158. Organization of Autocoder Assembly

3. Phase I: instructions and controls of the first phase of assembly.
4. Phase II: instructions and control of the second phase of assembly.
5. Macro-instructions: a library of tested sequences of 705 instructions. Each sequence is identified on the tape by a name.
6. Subroutines: a library of tested 705 subroutines. Each routine is also identified on the tape by its name.

During assembly, the 705 operates upon the instruction entries, one at a time, as input data. The entire process is under control of the autocoder system. A complete 705 program is produced as output with instructions originally written in autocoder language converted to five-character absolute machine coding.

The output program is written on tape together with additional instructions needed to load the program into the machine for operation. The program may also be punched in cards if desired (Figure 159).

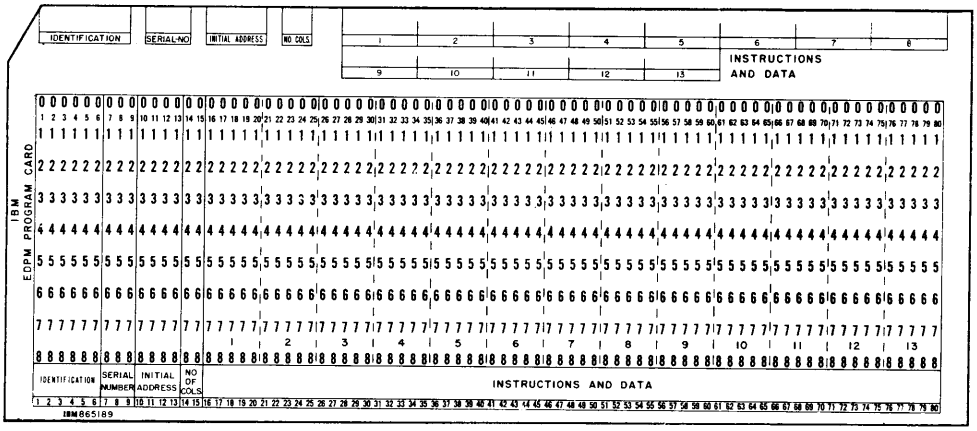


Figure 159. Output Program Card, Autocoder

A listing is written on tape for reference. It includes all entries exactly as originally produced by the programmer (Figure 160). The listing also includes the corresponding absolute coding for each entry as developed by the autocoder. The listing may be obtained as direct output on the printer at the time of assembly.

```
PG LINE    TAG          OP       NUM    OPERAND              PROGRAM TBL SH COMMENTS          LOC OP ASU ADDRESS
01 000                  TITLE                          SERIAL TABLE SEARCH

01 010 READ RCD         RDTP      2    RCD AREA¤EOJ RT¤¤
       READ RCD         SEL            0202                                          00164 2     00202 0202
                        RD             RCD AREA                                      00169 Y     00400 0400
                        DOA            TPERR
                        INCL           TPERR
       ¤¤000001         LOD       14   ¤¤000001                                      00174 8 14  00174 0AP4
                        TRA       14   TPERR                                         00179 I 14  00574 0EP4
                        RACON          EOJ RT                                        00184 A     00334 0334
                        RACON                                                        00189 A
                        INCL           XOFF
01 020                  RAD        4   #&0007#              TO STEP COMPARE ADDRESS  00194 H   4 01059 1 59
01 030                  LRL14          LAST CODE&7
       ¤0000002         LOD       14   ¤0000002  &5                                 00199 8 14  00204 0B-4
                        RACON          LAST CODE &7                                  00204 A     00562 0562
01 040                  MOVEI          BASIC ADDR¤TBL SEARCH¤
       ¤0000003         RCV       14   TBL SEARCH&1                                 00209 U 14  00221 0BK1
                        TMT       14   BASIC ADDR&1                                  00214 9 14  00566 0E06
01 050                  RAD            ITEM CODE                                     00219 H     00402 0402
01 060 TBL SEARCH       CMP                                  *ASU04                   00224 4
                        ADM        4   @00224                                        00229 6   4 00224 0S24
01 070                  TRE            ITEM FOUND                                    00234 L     00254 0254
01 080                  CMP       14   TBL SEARCH           IS THIS LAST ITEM        00239 4 14  00224 0BK4
01 090                  TRE            NO CODE              YES                      00244 L     00394 0394
01 100                  TR             TBL SEARCH           CONTINUE TABLE SEARCH    00249 1     00224 0224

01 110 ITEM FOUND       MOVEI          TBL SEARCH¤COMPUTE¤
       ITEM FOUND       RCV       14   COMPUTE   &1                                  00254 U 14  00271 0BP1
                        TMT       14   TBL SEARCH&1                                  00259 9 14  00221 0BK1
01 120                  LOD       14   #I998#               TO DECREMENT BY 2        00264 8 14  01068 1&08
01 130                  ADM       14   COMPUTE                                       00269 6 14  00274 0BP4
01 140 COMPUTE          RAD                                  UNIT COST                00274 H
01 150                  ST             UNIT COST                                     00279 F     00406 0406
01 160                  MPY            ITEM QTY                                      00284 V     00411 0411
01 170                  RND            1                                             00289 E     00001 0001
01 180                  ST             TOTAL COST                                    00294 F     00419 0419
01 190                  WRTP       1   RCD AREA¤¤¤
                        SEL            0201                                          00299 2     00201 0201
                        WR             RCD AREA                                      00304 R     00400 0400
                        DOA            TPERR
       ¤¤000002         LOD       14   ¤¤000002                                      00309 8 14  00309 0C-9
                        TRA       14   TPERR                                         00314 I 14  00574 0EP4
                        RACON                                                        00319 A
                        RACON                                                        00324 A
01 200                  TR             READ RCD                                      00329 1     00164 0164

01 210 EOJ RT           WRTM       1
       EOJ RT           SEL            0201                                          00334 2     00201 0201
                        WTM                                                          00339 3     00001 0001
                        DOA            TPERR
       ¤¤000003         LOD       14   ¤¤000003                                      00344 8 14  00344 0CM4
                        TRA       14   TPERR                                         00349 I 14  00574 0EP4
                        RACON          EOJ RT    &30                                 00354 A     00364 0364
                        RACON                                                        00359 A
01 220                  RWD                                                          00364 3     00002 0002
01 230                  TYPE           #END OF JOB¤#
                        SEL            0500                                          00369 2     00500 0500
                        WR             #END OF JOB¤#                                 00374 R     01069 1069
                        DOA            XOFF
       ¤¤000004         LOD       14   ¤¤000004                                      00379 8 14  00379 0CP9
                        TRA       14   XOFF                                          00384 I 14  01019 1&J9
01 240                  HLT            9999                                          00389 J     09999 9999
01 250 NO CODE          HLT            1                                             00394 J     00001 0001
01 260                  TR             NO CODE                                       00399 1     00394 0394

02 010                  DRCD                                                         00400
02 020 RCD AREA            1                                                         00400
02 030 ITEM CODE           2   &                                                     00402
02 040 UNIT COST           4   &                                                     00406
02 050 ITEM QTY            5   &                                                     00411
02 060 TOTAL COST          8   &                                                     00419
```

Figure 160. Autocoder Program Listing

PROGRAM INDIAN PROBLEM IDENT. SOL 1

PG	LINE	TAG	OPERATION	NUM.	OPERAND
	01		SEL		100
	02		RD		DATA CARD
	03		RAD	1	YEARS
	04	CALCULATE	RAD		RATE
	05		MPY		PRINCIPAL
	06		RND		2
	07		ADM		PRINCIPAL
	08		SUB	1	ONE
	09		TRZ	1	PRINT
	10		TR		CALCULATE
	11	PRINT	RAD		PRINCIPAL
	12		SPR		ANSWER
	13		SEL		500
	14		WR		ANSWER
	15		HLT		9999
	16	DATA CARD	DRCD		
	17	RATE		2	+
	18	PRINCIPAL		12	+
	19	YEARS		3	+
	20	ONE		1	+
	21	ANSWER		18	
	22	GM		1	

Figure 161. Autocoder Solution 1, Indian Problem

Basic Autocoder Programming

Figure 161 is a program for the Indian Problem described in previous sections as it might be written in Autocoder. The program is divided into two types of entries: (1) instructions (lines 01-15), and (2) a description of the input record area (lines 16-22).

Record Area

The first entry of the record area (line 16) identifies the entire input record by a descriptive name referred to as a "tag." The tag may be as many as ten characters in length and is written here as data card. The mnemonic operation in this entry is DRCD, meaning "define the record."

Note that DRCD is not a 705 operation. It is used only by the autocoder assembly process to specify that entries following the DRCD describe fields or other portions of data input or output. These following entries will be used to assign space in memory for the various records described. Their tags can be used as operands of instructions referring to the data.

Each field of the data card is therefore tagged with an appropriate name and its length is indicated in the numerical column of the program sheet. The programmer also indicates whether the fields are to be signed. A total of 37 memory positions are reserved, including one for a group mark.

Instructions

The instructions in the autocoder program are identical with those shown in Figure 118 except that they are written in autocoder language. The select instruction on line 01 selects the card reader. The read instruction on line 02 is addressed to "data card," the tag of the input record.

An instruction may be addressed to the tag of another instruction. For example, the transfer instruction is addressed to "calculate." The transfer on zero instruction is addressed to "print."

Instructions may carry the actual address of machine units as shown by the operands of select, round and halt on lines 01, 06, 13, and 15. Or, they may have "literal" operands consisting of the literal data to be operated upon. This type of operand is shown in a second example.

Constant Area

Figure 162 shows a second method of writing the program originally shown in Figure 118. In this example the data card is not needed. Instead, the data are assembled as part of the completed output program and are loaded into memory with that program.

Data are given in two ways: (1) as constants in a separate area reserved for this purpose, and (2) as literal operands in the instructions.

For example, the program entry on line 13 is the operation DCON, meaning "define constant." Entries following this operation are included as a portion of the completed output program. Each constant entry is tagged for reference by the instructions. The length of the constant is indicated in the numerical column and the constant is written in the operand columns

PROGRAM INDIAN PROBLEM IDENT. SOL 2

PG	LINE	TAG	OPERATION	NUM.	OPERAND
	01		RAD	1	(+332)
	02	CALCULATE	RAD		PRINCIPAL
	03		MPY		(+03)
	04		RND		2
	05		ADM		PRINCIPAL
	06		SUB	1	(+1)
	07		TRZ	1	PRINT
	08		TR		CALCULATE
	09	PRINT	RAD		PRINCIPAL
	10		SPR		ANSWER
	11		WR		ANSWER
	12		HLT		9999
	13		DCON		
	14	PRINCIPAL		12	+000000002400
	15	ANSWER		18	$b,bbb,bbb,bbb-bbb
	16	GM		1	‡

Figure 162. Autocoder Solution 2, Indian Problem

exactly as it is to be used. If the field is signed, the proper sign is indicated as shown.

Instructions for Indian Problem

In Figure 162, the first instruction on line 01 has an operand written as (+332). The enclosure of a field by parentheses indicates to the assembly that the characters so enclosed are to be treated as data. In this case, the quantity 332 is to be placed in ASU 01 by a reset-and-add operation. Plus 332 is the literal amount to be added.

Literal operands are used on lines 03 and 06. This feature makes it unnecessary for the programmer to establish a location for constants in memory or to define them with a DCON operation. He may insert literal data as the need arises. The assembly locates the constants for him and generates the proper addresses which then refer to the data.

Additional advantages of the autocoder may now be summarized.

1. The program is written in a notation referring to records, data, or instructions by name rather than code.

2. Data may be entered as they are used, saving clerical time in establishing tables of constants that must be referred to by absolute memory location. The location of the constant data and the conversion of instruction operands to addresses are automatic features of the assembly.

3. The programmer may also use actual memory locations if desired. To do this, the operand portion of the autocoder instruction is preceded by the symbol @. For example, RAD @ 1234 refers to an actual memory position.

4. Other special mnemonic operations may be used to insert titles, headings, descriptions and other descriptive information where needed.

The entire Autocoder system is more fully explained in the *IBM 705 Autocoder System Manual of Operation,* Form 222-6726.

Index

IBM®

International Business Machines Corporation
590 Madison Avenue, New York 22, N.Y.

Printed in U.S.A. D22-6509-0