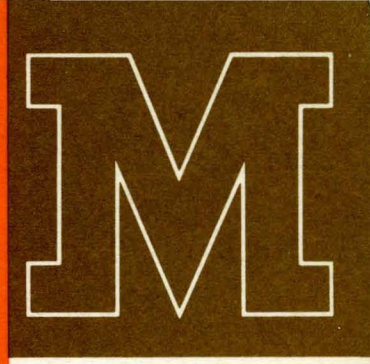


IBM

General Information Manual

IBM 709-7090 Data Processing Systems

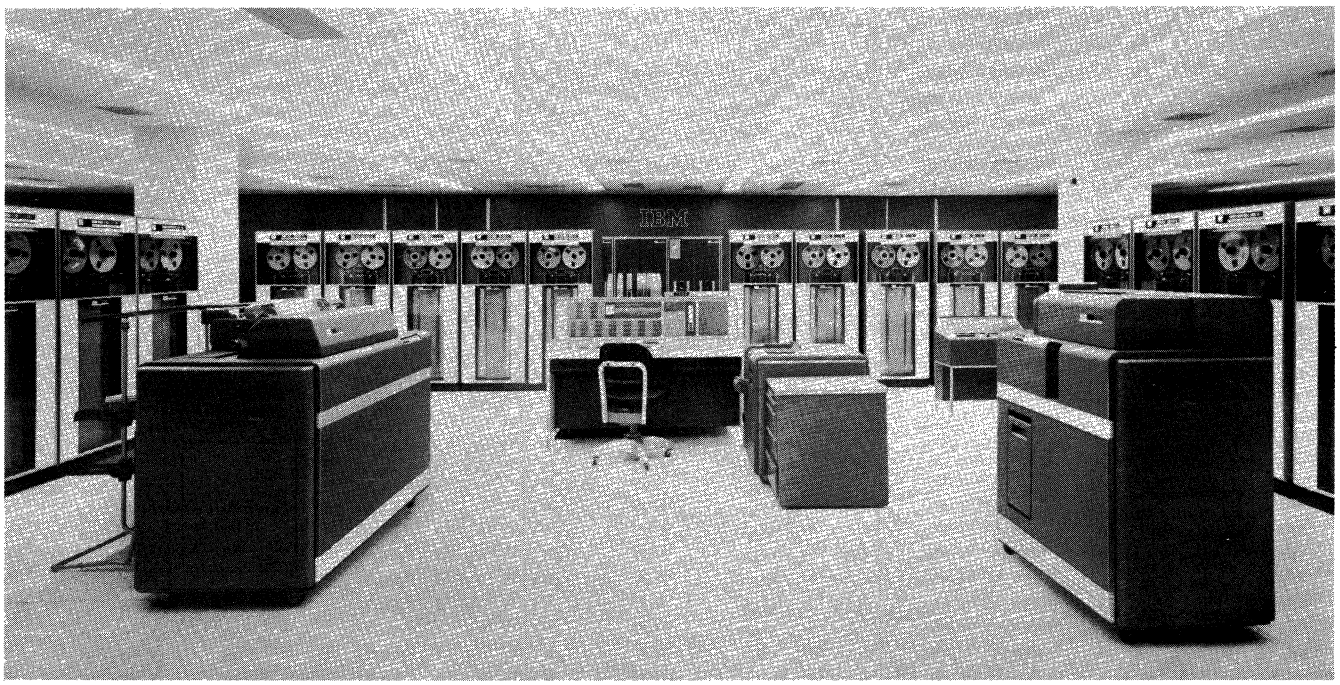
IBM

General Information Manual

709-7090 Data Processing System

MINOR REVISION (August, 1960)
This edition, Form D22-6508-2, is a minor revision of the preceding edition but does not obsolete Form D22-6508-1. The principal change in this edition is the addition of a section on 1401 operation on page 31.

Contents

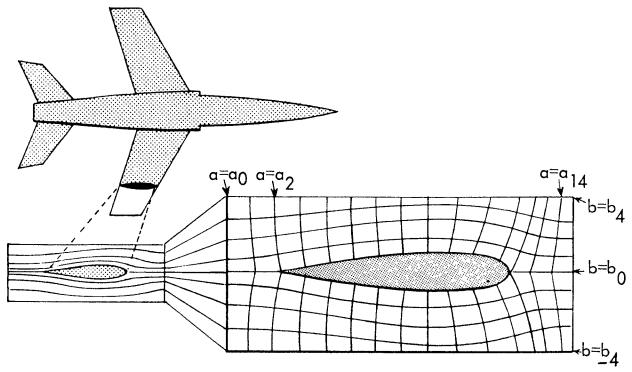


IBM 7090 Data Processing System

Data processing systems are finding new application in virtually every phase of science, business, and industry. Rapidly expanding scientific investigations involve many complex calculations. The vast amount of data constantly being used in aircraft industries, government agencies, and business establishments of all kinds demand machines to compute, select, and correlate data at electronic speeds.

Figure 1 shows the kind of problem that computers are solving for scientists today. Many similar problems are encountered and solved in the scientific field; however, the computer is adaptable to business applications as well. The computer can be, and is, used for payroll, material inventory, or any of a great number of other business problems requiring prompt, accurate results.

In the sample problem, the computer calculates the density and velocity of air at each of the mesh-points shown. From these data, engineers can evaluate the lifting power and drag of the wing section. By solving a number of these problems, an optimum new wing design is developed.



The mathematical formulas used are:

$$1.\ \rho V_a \frac{dV_a}{da} + \rho V_b \frac{dV_a}{db} + \frac{d}{da}(pg^{aa}) = 0$$

$$2.\ \rho V_a \frac{dV_b}{da} + \rho V_b \frac{dV_b}{db} + \frac{d}{db}(pg^{bb}) = 0$$

$$3.\ \frac{d}{da}(\rho V_a) + \frac{d}{db}(\rho V_b) = 0$$

$$4.\ \quad p = \int(\rho)$$

Figure 1. Aircraft Application

Equations 1, 2 and 3 are replaced by difference equations over a network as shown in the figure. This difference system and equation 4 form a set of simultaneous non-linear algebraic equations. These are solved on the computer by repeating calculations, called an iteration method. In this one typical problem, there are 800 equations, 100 iterations, 80,000 operations per iteration, and 8,000,000 operations per solution.

Figure 2 shows a comparison, in time, between manual methods of solution and solution by computers. The times are approximate. This comparison shows that computers of today now have the ability to solve problems that cannot be solved in a lifetime of manual labor.

One of the first things to be considered in a computer system is an understanding of the functions of its various components. A simplified computer system consists of three main units as shown in Figure 3.

In a typical computer system, several types of input-output devices are needed. These may be units such

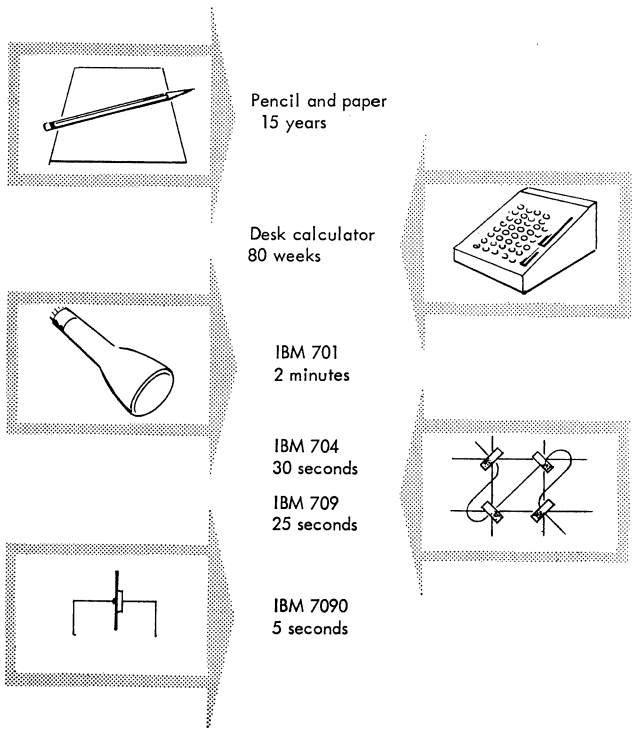


Figure 2. Time Comparison of Computation Methods (Approximate)

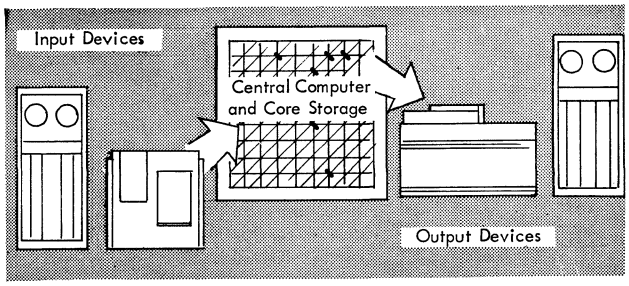


Figure 3. Main Computer Components

as punched card readers, a device to read or write on magnetic or paper tape, a telephone or teletype line, punched card recorders, and printing equipment.

An input component is any device capable of feeding information into a computing and data handling unit, usually called the central processing unit.

A storage unit must also be provided to store or "remember" all input data and instructions (also called orders) to the computer. These instructions are then used to tell the computer how the processing of data is to be executed.

The central processing unit has an arithmetic section to accomplish all arithmetic and logical operations and a section to modify or change instructions to adapt the processing to variations within the established procedure. A control section is also a part of the central processing unit and directs the computer in its operation by making logical decisions. An example of one of these decisions would be a test to determine if a number is positive or negative. As a result of this test, the computer can take a different processing course for each alternative. Simple tests may be combined to make complex logical decisions.

The output components are any devices capable of recording the resultant data after the central processing unit has operated upon it.

Binary Notation

The common decimal notation of the commercial and scientific world is a familiar one. This notation is so familiar that its use is hardly questioned. However, it is possible that, for some purposes, other numbering systems are more convenient.

What numbering system to use is entirely a matter of convenience. Decimal notation is used because it is most familiar and is understood by most people. However, had our primeval ancestors developed eight fingers instead of ten, we would probably be more familiar with a numbering system based on eight, rather than ten, and we might consequently question the decimal system.

The decimal system, with its ten digits, is learned by most people early in their training. This system serves well for counting purposes. Why then should computers designed to assist mathematicians, engineers and businessmen, be designed to use the binary system of numbers?

The reason is that current digital computers use binary circuits; therefore, the mathematics of computers is binary in nature. The octal system is a shorthand method of writing long binary numbers. Octal notation is used when discussing the computer, but has no relation to the internal computer circuits.

The binary, or "base two" system, uses the two symbols 0 and 1 to represent all quantities. Counting starts in the same manner as in the decimal system with a 0 for zero and then a 1 for one. At two, however, there are no more symbols to be used. It is therefore necessary to take the same step at two in the binary system that is taken at ten in the decimal system. This step is to place a 1 in the next position to the left and start again with a 0 in the original position. A binary 10 is equivalent in this respect to a 2 in the decimal system. Counting is continued in an analogous manner with a carry to the next higher order every time a two is reached instead of every time a ten is reached. Counting in the binary system is as follows:

Binary				Decimal	Binary				Decimal
8	4	2	1		8	4	2	1	
			0	= 0				1	= 1
		1	0	= 2			1	1	= 3
	1	0	0	= 4		1	0	1	= 5
	1	1	0	= 6		1	1	1	= 7
1	0	0	0	= 8	1	0	0	1	= 9

Octal Notation

It has already been pointed out that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. This is not a problem to the computer itself, but in talking and writing, these binary numbers are bulky. A long string of ones and zeros cannot be effectively transmitted from one individual to another. Some shorthand method is necessary. The octal number system fills this need. Because of its simple relationship to binary, numbers can be converted from one system to another by inspection. The base of the octal system is 8. This means there are eight symbols: 0, 1, 2, 3, 4, 5, 6, and 7. There are no 8's or 9's. The important relationship is that three binary positions are equivalent to one octal position. The following table combines what has been shown concerning decimal and binary with the octal numbers.

BINARY	OCTAL	DECIMAL
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7

At this point a carry to the next higher position of the number is necessary, since all eight symbols in the octal system have been used.

1000	10	8
1001	11	9
1010	12	10
1011	13	11

As far as the internal circuitry of the computer is concerned, it only understands binary ones and zeros. The octal system is used to provide a shorthand method of reading and writing binary numbers so that the contents of a register, when shown on the operator's panel, may be read directly. This is shown in Figure 4, using a 36-digit binary number.

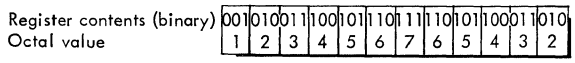


Figure 4. Binary Representation of Register Contents

Magnetic Cores

The main storage medium for many computers is the magnetic core.

Each magnetic core is a ring or doughnut shaped piece of ferromagnetic material. The cores "remember" information indefinitely, and can recall it in a few millionths of a second. When a wire is inserted through the hollow center of a core (Figure 5), an electrical current passed along the wire sets up a magnetic field around the wire. This field magnetizes the core. When the current is removed, the core remains magnetized. If a current is passed along the wire in the opposite direction, the magnetic field set up around the wire is reversed. When this occurs, the core is said to have "flipped" or changed its magnetic state. A "sense" wire is inserted through the core and, when the core flips, a small electrical voltage is sent along the sense wire. This voltage may then be amplified and used in the computer (Figure 6).

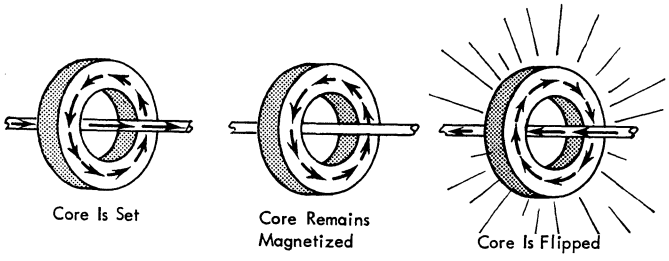


Figure 5. Magnetic Core Action

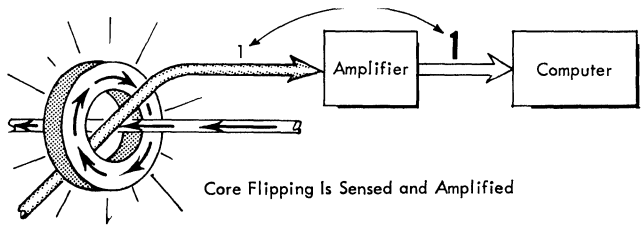


Figure 6. Flipping of a Magnetic Core

As with the magnetic core, all computer elements are able to represent two states. These two distinct states provide the basis by which these elements hold information. For this reason, the elements are called *bi-stable* elements. For example, one state may be interpreted as the digit 0; the other, as 1. Similarly, the elements may be used to represent: plus or minus, on or off, yes or no, and so on. Several of these elements are shown in Figure 7.

The IBM 709 and 7090 Data Processing Systems use high-speed core storage units. Each unit is divided into distinct sections called locations. Each location is uniquely identified by a number assigned to it. This identifying number is called an *address* because just as a street address denotes the precise location of a particular building on that street, this number de-

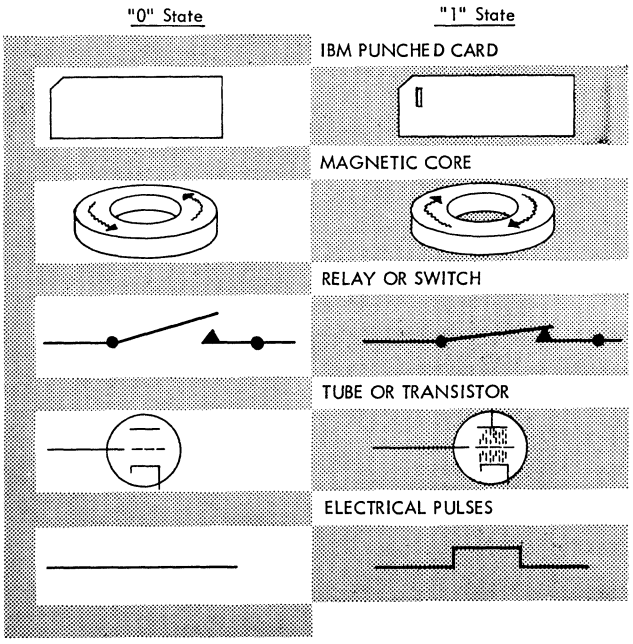


Figure 7. Bi-stable Computer Elements

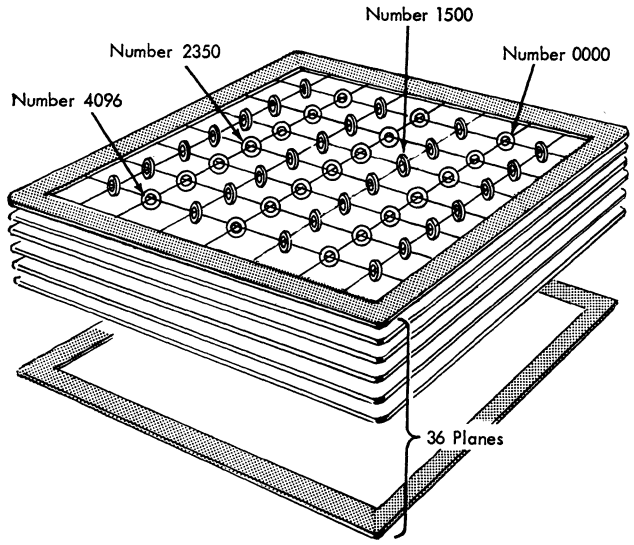


Figure 8. Core Storage Unit

notes the location of a particular item of data inside the core storage unit called a "word" (Figure 8).

Each word is further subdivided into elements called *bits*. Each bit has a value of either a 0 or a 1. Thus, the bits (36 in each word) are the basic units of information in the computer. Figure 8 also shows 36 "planes" or "stories" in the core storage "building." The operation of locating a word in core storage may be compared to the operation of an elevator in an office building. The elevator picks up passengers from each floor; in core storage, one bit of information is read or stored from each plane. In actual computer operation, all 36 bits are read, stored, or operated upon simultaneously, one from each plane.

As previously stated, one word contains 36 bits of information. Core storage contains two types of words:

1. A word upon which arithmetic or logical operations are to be performed is called a *data-word*.
2. A word interpreted by the computer as a code to "order or instruct" it to perform a particular operation is called an *instruction-word*.

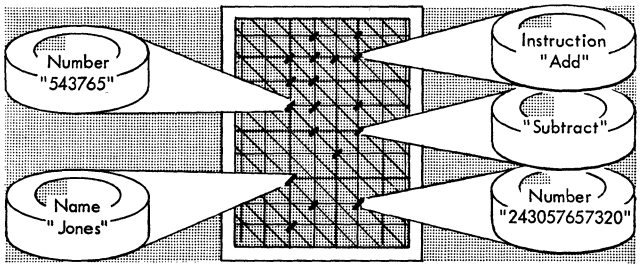


Figure 9. Instructions and Data in Core Storage

Both types of words are combinations of 36 zeros and ones. An instruction is able to "direct" the computer to perform some type of operation, e.g., read, write, add, subtract, or test for zero. If a data word were incorrectly used as an instruction, the computer might perform an illegal operation depending upon the data bit configuration. Data words form records, fields, amounts, results, and so on. Figure 9 is a schematic of core storage with both types of words contained in separate locations.

Core storage units are available to provide the computers with a capacity of 4096, 8192, or 32,768 word locations of 36 information bits each. As a decimal digit is expressed by three of these bits, a total of 11 significant decimal digits may be expressed by each word. In BCD coding (alphamerical characters), one word contains six numerical or alphabetic characters. Thus, the 32,768 words of core storage may contain the binary equivalent to 360,448 decimal digits of storage or the BCD representation of 196,608 characters. The actual capacity of storage is thus directly related to the type of coding used.

Figure 10 shows the three core storage units that are available for the 709 and 7090 Data Processing Systems. The IBM 737 Core Storage, used with the 709 system, has 4,096 words of storage. The IBM 738 and the IBM 7302 Core Storage have 32,768 words of storage and are used with the 709 and the 7090 systems, respectively.

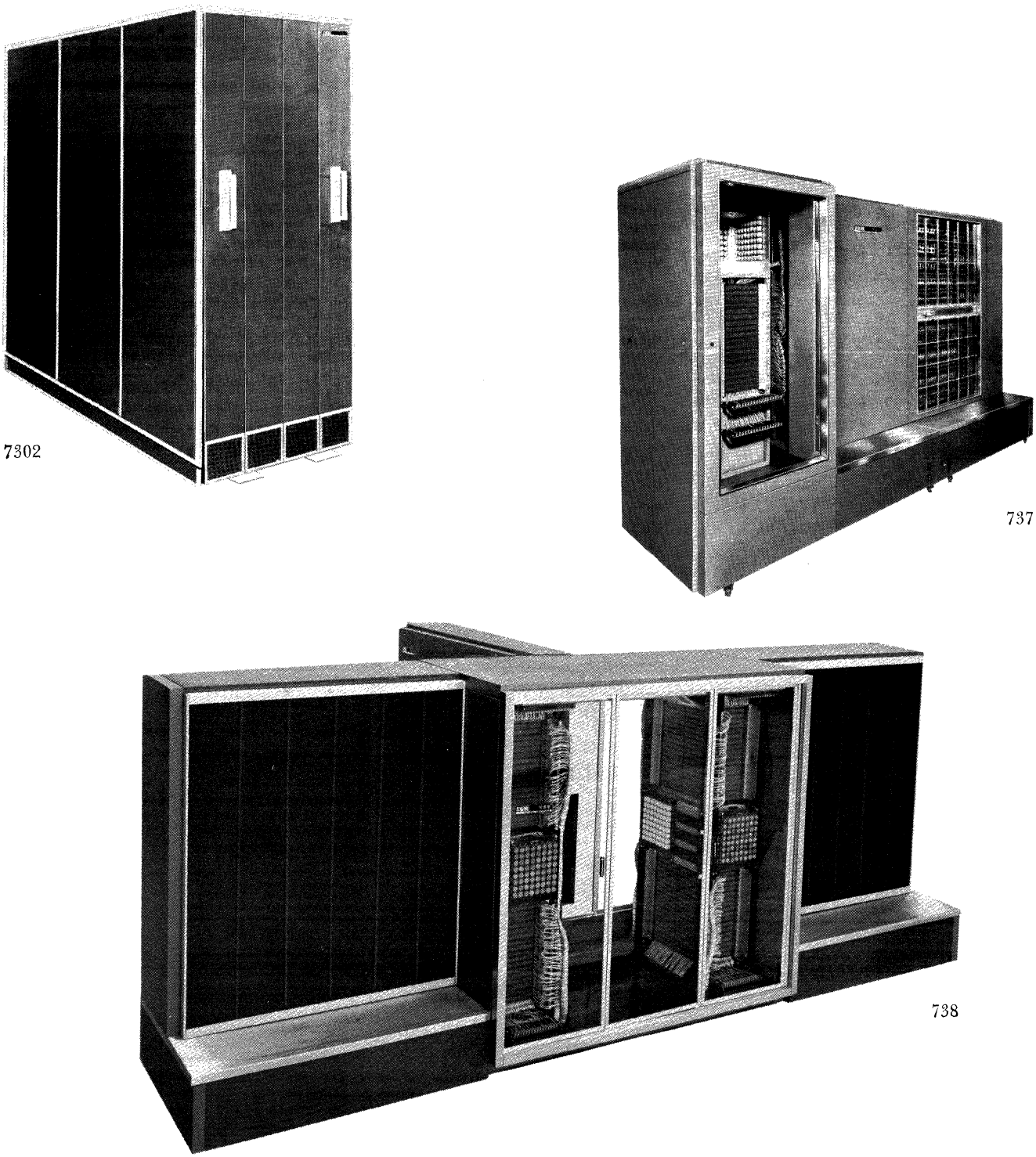


Figure 10. IBM Core Storage Units

Central Processing Unit

The central processing unit can be divided into two basic parts, information processing elements and execution controls. The information processing elements are normally referred to as the arithmetic section. The execution controls are called simply the control section.

The arithmetic section is the computer's problem-solving unit. The operations of addition, subtraction, multiplication, and division, as well as shifting, transferring, and storing of results are executed in this section. Figure 11 shows a few of the operations performed in the arithmetic unit.

The control section guides the computer through the operations necessary to complete the various instructions.

The execution of a group of instructions may be compared to compounding a chemical formula. The arithmetic section would be the laboratory where the compounding takes place; it would call on core storage for materials needed to produce the result. The control section would follow directions given in the formula. It would instruct the chemist as to what ingredients to mix, how long to mix them, and in what order they should be combined.

Of necessity, then, the control section contains a device called the "instruction sequencer." This device locates the proper instruction to be executed; then, while the instruction is being executed, this device sets up the conditions for the next instruction. At different times in the program, the next instruction to be executed may depend on the result of a test instruction, for example, whether an error indicator is on or off (Figure 12).

The arithmetic operations which computers are capable of performing treat each part of a word in

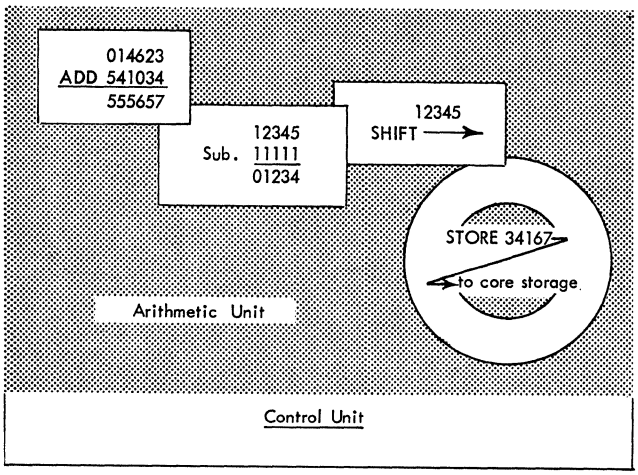


Figure 11. Arithmetic Operations

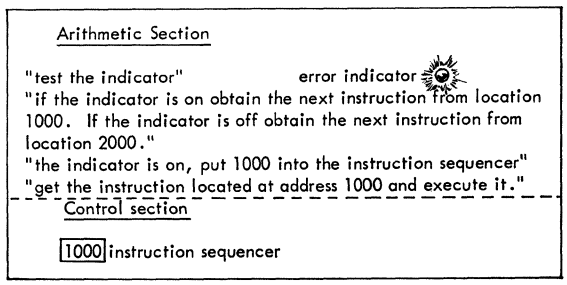


Figure 12. Conditional Testing and Transferring

core storage as either a 1 or a 0. Thus, a computer word consists of 36 zero and one digits and is said to be expressed in the "binary system." An example of a binary word as it appears in the arithmetic unit is 001101110010011011001001111001001001 which is simply a string of zeros and ones. By doing its arithmetic in the binary system, the computer is able to achieve greater speeds and efficiencies than would otherwise be possible.

Normally, in solving a problem, the numerical data are originally entered into the computer in decimal or IBM card coding. The computer is then instructed to translate the data into binary form, go through the desired computations, and then translate the results back into decimal form (Figure 13). However, data may be entered into the computer in any form desired by the programmer.

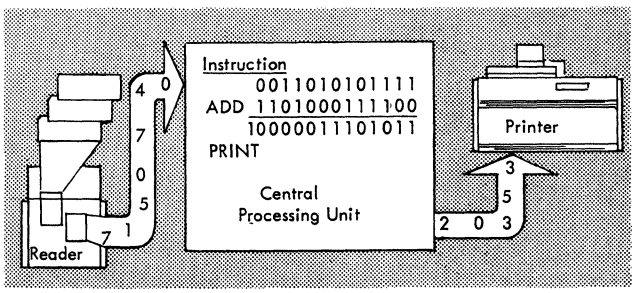


Figure 13. Decimal and Binary Data Flow

Stored Program

The work accomplished by the computer in solving a problem or processing data consists of executing many instructions at high speed. To solve a problem, it must first be reduced to equations or instructions that the computer is capable of performing. The entire set of instructions used in solving a problem form a *program* for the computer. Because these instructions are held in the computer's storage unit, it is called a *stored program* system.

Normally, instructions are taken from sequentially ascending locations of core storage. However, the execution of instructions does not necessarily have to occur sequentially. It is possible, when control or transfer instructions are given, for the computer to alter the process of sequential execution and to indicate some particular location in core storage containing the next instruction word to be executed. In this way, the execution of any instruction or block of instructions may be repeated as often as desired.

For some control instructions, whether the next instruction is taken in sequence or from some other specified location may depend on the result of a test (Figure 12). In this case, the control operations provide the program with decision-making abilities. The logical path followed by the program (that is, the precise sequence of instructions to be executed) may be controlled by a series of tests applied at various points in the program. Doing this gives a stored program the ability to change its course of execution. These conditional operations increase immeasurably the scope of the system's application.

Most computer instructions have an address part indicating the location in core storage subjected to some arithmetic or logical operation. This address part, or field, always occupies bit positions 21 through 35 (Figure 14) in an instruction word.

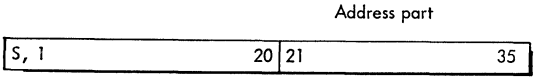


Figure 14. Address Part of an Instruction

The 15-bit address field is large enough to hold the number 32,767 — the largest address in core storage. This number expressed in the binary system is simply 15 consecutive ones (Figure 15).

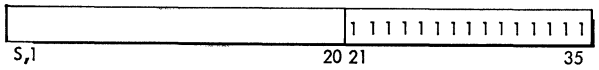


Figure 15. Address Part Showing Maximum Address

In a data processing system with 8,192 words of core storage, the largest address (8191) is contained in only 13 bit positions of the instruction word. The contents of the two left-most positions of the address field are ignored during the execution of an instruction to decode an address. Similarly, a computer with 4,096 words of core storage uses only bit positions 24 through 35 as its address field.

The operation part of an instruction normally is not fixed in length but may vary depending upon the instruction itself. Figure 16 shows the bit pattern for an instruction specifying the addition of the contents

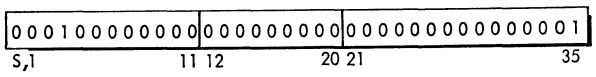


Figure 16. Add Instruction

of core storage location 0001. When this instruction is executed by the computer, the contents of core location 0001 are added to the contents of the accumulator register.

For example, assume that the accumulator contains the number +1. If the number in location 0001 is +2, the result of executing the ADD instruction (Figure 17) is a 1 bit in positions 34 and 35, and 0 bits in all other positions of the accumulator; (11 is the binary expression for the sum, 3). The first position of the word is used to express the sign of the amount or result.

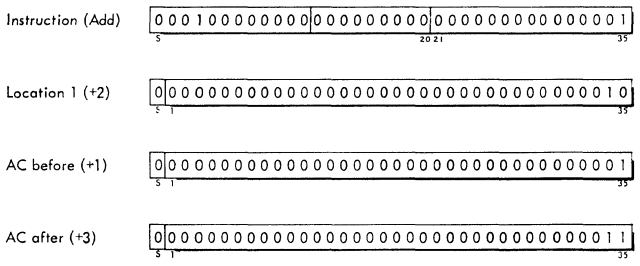


Figure 17. Execution of an Add Instruction

A register is defined as a device with the ability to accept data, hold them, and transfer the data to another register or device. The registers are given different names according to their functions. Thus, the accumulator register "accumulates" arithmetic results. The multiplier-quotient register holds either the multiplier or the quotient in an arithmetic operation. On all arithmetic operations, the sign positions of the registers in use are automatically adjusted.

The accumulator register has 38 positions. They allow the register to handle a 35-bit word with its sign. Two extra positions, P and Q, are provided to keep track of overflow conditions. If two 35-bit numbers are added together, it is possible that the result would be larger than 35 bits, as shown in Figure 18. The accumulator is used to hold one factor during arithmetic operations. The other factor usually is

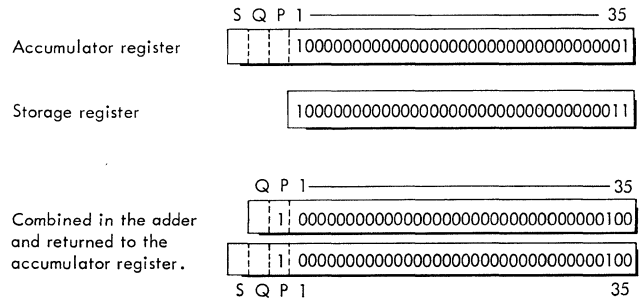


Figure 18. Accumulator Overflow Condition

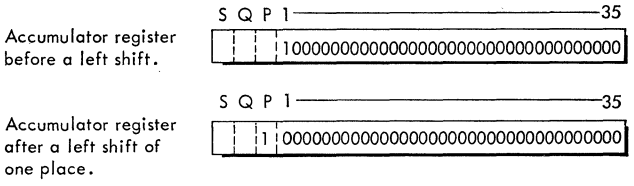


Figure 19. Accumulator Shifting

held in the storage register. The two factors are combined in another register-type device called the adder. The adder consists of 35-bit positions and the two extra ones, P and Q. Thus, any overflow out of position 1 will be placed in position P. Likewise, any overflow out of position P will be placed in position Q. An overflow out of position Q will, however, be lost.

Information in the accumulator may be shifted to the right or to the left. If a left shift is involved, an overflow condition may be recorded in the same manner as in an arithmetic operation (Figure 19). Not only is the entire number shifted to the left, but zeros are inserted (position 35) in the positions vacated. Thus, no matter what the content of the accumulator, if a left shift of 37 or more places is executed, the contents of the accumulator are replaced by zeros.

Information may also be shifted into the accumulator from the MQ register, one bit at a time, or from the accumulator to the MQ, one bit at a time. The effect is to create a register that is 74 positions in length, 38 positions for the accumulator and 36 for the MQ register.

The multiplier-quotient (MQ) register has 36 positions. During a multiply operation, this register contains the multiplier; during a divide, it receives the quotient. The register can store and shift a full 36-bit word. In addition to shifting right or left, in the same manner as the accumulator, it can also "ring shift." That is, bits shifted out of the sign position enter position 35 (Figure 20).

In addition to its arithmetic and shifting functions, the MQ register serves as a sending and receiving register for some input-output operations. For the 709 system, this means that information coming from the magnetic drum or going to the drum and CRT equipment passes through the MQ register.

When a word is entered into or taken from a location in core storage, *a storage reference* is said to be made. This operation is non-destructive. That is, the contents of that location are left unaltered after the operation. However, if the storage reference causes new information to be entered into a location, the prior contents of that location are automatically replaced by the new information.

Only one such storage reference can be made during any single computer cycle. The basic computer cycle (or internal speed) for the 709 is 12 microseconds long (12 millionths of a second). The basic cycle for the 7090 is 2.18 microseconds.

Normal computer operations start with an instruction (I) cycle. This cycle does the following:

1. It obtains the instruction to be executed from the core location designated by the instruction counter.
2. It locates the operand (number to be worked with or upon), if any, as specified by the instruction's address part.

An example would be the instruction ADD 0002 (add the contents of storage location 0002 to the contents of the accumulator register), shown in Figure 21. All data handling concerned with storage and the central processing unit is accomplished in parallel. This means that all operations are concerned with a full word of 36 bits.

As an example, refer to Figure 17 where an ADD operation is performed. Even though the numbers involved use only a few bits of the entire word, the remainder of the word is filled with zeros and all 36 positions of both words are then added in one operation. Thus the time required to add two 1-bit numbers is the same as the time required to add two 36-bit numbers.

This principle of parallel operation is one of the basic differences between the 709 and 7090 computers when compared with the 705 system series, which is said to operate serially (normally one position or character at a time).

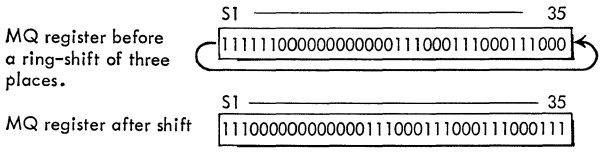


Figure 20. MQ Register "Ring Shift"

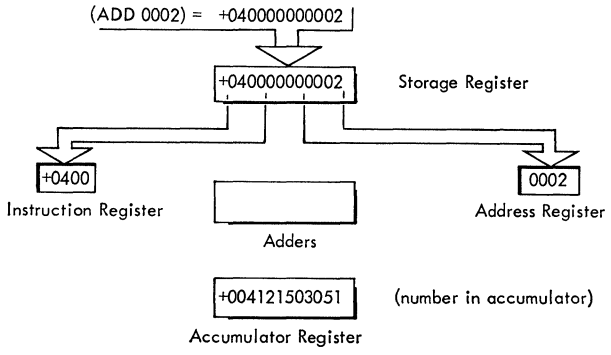


Figure 21. Schematic, Decoding an Add Instruction

The instruction ADD 0002 is brought from core storage into the *storage register*. At this point, the operation part is sent to the *instruction register* to determine the type of operation to be executed. At the same time, the address portion of the storage register is taken to the *address register* so that the proper core storage location (operand) is added during the next storage reference cycle. All numbers in Figure 21 are shown in octal notation.

The next cycle, referred to as an execution cycle, is used when a reference to core storage is required to obtain another word, e.g., a number to be added. Assume that the address register has the address of the core storage location to be added. During this second (reference) cycle, the contents of that storage location are brought from core storage to the storage register, and then to the *adders*. At the same time, the previous number in the *accumulator register* is taken to the adders. A new number is formed in the adders and is then placed back in the accumulator register (Figure 22).

The operation is now finished and the computer is ready to execute the next sequential instruction.

The type of operation just described is called fixed point arithmetic or integer arithmetic. The computer can also perform operations in floating point arithmetic. A complete set of floating point instructions is provided to increase the range of numbers used and to reduce programming time.

In integer arithmetic, the size of the numbers used is fixed by the design of the computer (36-bit word size). By using floating point arithmetic, a larger number may be expressed and operated upon, because a part of the word is used to express the exponent (characteristic) and another part of the word is used for the fraction (mantissa).

A comparison of number size shows that the largest number the computer can use with fixed point operation is 1×10^{11}. This is equal to 100 billion. With floating point operation, however, 3×10^{35} is the largest number. This number is too high to be expressed in common language terms.

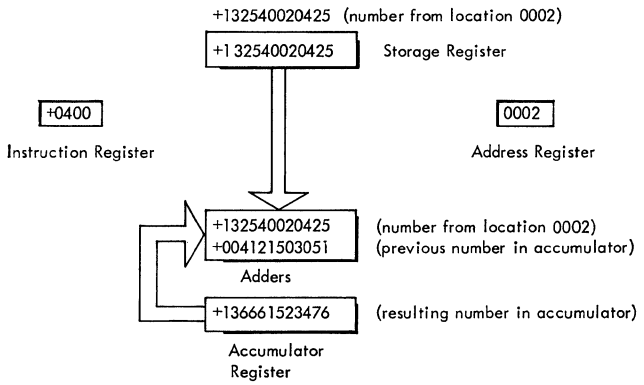


Figure 22. Schematic, Performing an Addition

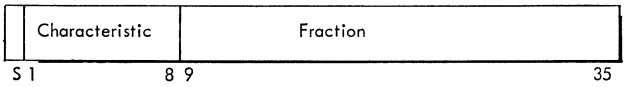


Figure 23. Floating-Point Word Format

In the computer, a floating-point number is stored in a word as shown in Figure 23. The fraction is contained in bit positions 9 through 35 with the sign of the fraction contained in position S of the word.

The characteristic is contained in bit positions 1 through 8 and is formed by adding +128 to the exponent. For example, an exponent of —32 would be represented by a characteristic of 128—32 or 96. An exponent of +100 would be represented by a characteristic of 100+128 or 228.

Most integer arithmetic instructions have a floating-point counterpart instruction. Thus, it is possible to ADD or to perform a floating-point ADD (FAD).

To summarize, a floating-point binary number (X) may be represented as a signed proper fraction (B) times some integral power (b) of 2. Examples:

X		B		2^b
−.001	=	−.100	×	2^{-2}
.100	=	.100	×	2^0
1.100	=	.110	×	2^1
110.000	=	.110	×	2^3

Assembly Programs

The writing of a complete program for the computer in its machine language would be rather awkward. For example, to write only the one instruction required to subtract the contents of core location 0003 from the contents of the accumulator register requires the recording of 36 zeros and ones, as shown in Figure 24.

This instruction can be written as SUB 0003, a more convenient form of expressing the instruction. The writing of instructions in this form is an improvement over the system of writing 36 zeros and ones. In addition to reducing the amount of information which must be written, it has more meaning to the reader. Namely, "subtract the contents of location 0003." However, to determine the entire machine operation, the programmer or person reading the program must

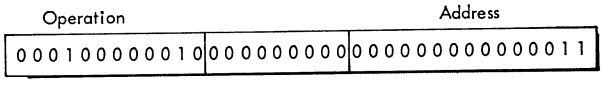


Figure 24. Subtract Instruction

know what is stored in location 0003. To assist the programmer, the use of symbols can be extended to include addresses as well as the operation part of an instruction.

Assuming that the net pay for a payroll calculation is stored in location 0003, the instruction might now be written as "subtract net." This clarifies to the programmer the operation to be performed, and also what quantity is involved. Thus, symbolic programming employs instruction-by-instruction coding in a language that is a representation of the basic language of the computer itself.

The computer can execute instructions only if they are stored in machine language. Thus, the instruction "subtract net" must be converted to the basic machine language before its execution. One way of doing this would be to manually convert all symbolic instructions to machine language before entering them into the computer. This obviously would be a laborious task. A more practical solution is to have the computer perform the conversion. This is accomplished through use of a *symbolic assembly program.*

An assembly program performs the necessary translation or conversion from the symbolic language to the machine language. At the same time, the program assigns absolute core storage locations to the instructions and data contained in the symbolic program. Figure 25 shows a flow chart of the process involved when using an assembly program. Assume that the instructions to be assembled are on punched cards and that the assembly program itself is on a magnetic tape (assembly tape).

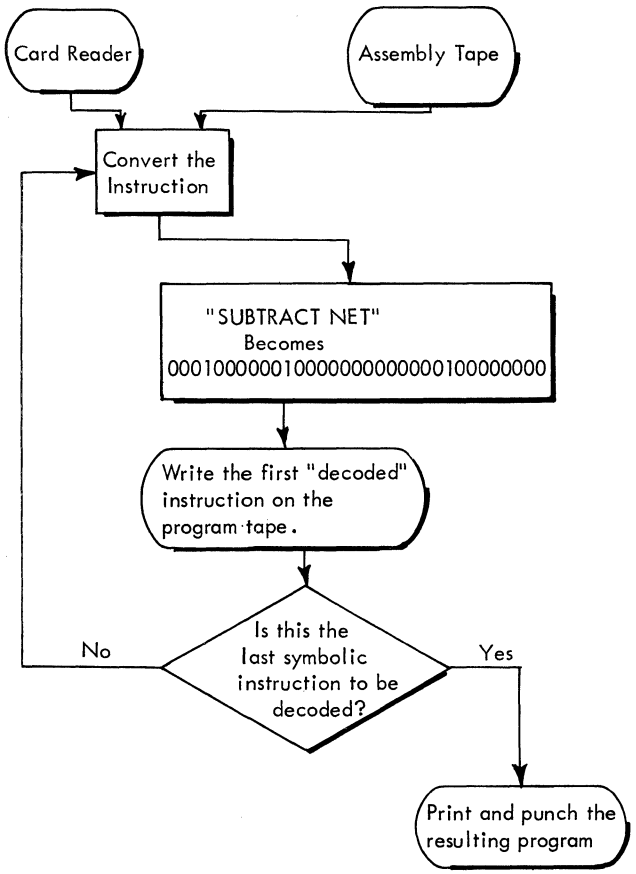


Figure 25. Flow Chart of an Assembly Program

The assembly program normally provides certain restrictions upon the length and type of symbols that may be used by the programmer. It assigns an absolute core storage location for the first symbolic instruction and increases this location by one for each subsequent instruction decoded. In this manner, the decoded program instructions are assigned sequential locations. Space is provided for the data that are a part of the assembled program. Normally, the constants and other data are placed at the end of the symbolic program and are assigned sequential locations that follow the instruction area.

During assembly, the first reference to a symbolic address assigns this address to an absolute storage location. Any subsequent references to this symbolic address will also be assigned the same address. The manner in which the assembly proceeds is described below.

Each symbolic instruction is read from the card reader into the central processing unit. The "subtract" portion of the first instruction is matched against the contents of the assembly tape and is decoded into the proper bit configuration for a subtract operation 000100000010. The "net" portion of the instruction is assigned a location in core storage that is not being used. This bit configuration (assume 000100000000) is then inserted into the address portion of the decoded subtract instruction. The full instruction word (000100000010000000000000000100000000), both operation and address parts, is then written as the first instruction on the program tape. The original "subtract net" is stored so that it may appear in the final printing of the program. The assembly program then tests to determine if more instructions remain to be decoded. If there are more, the above process is repeated until all symbolic instructions have been decoded and written on the program tape. When the assembly process is complete, the resultant assembled program is printed, together with the original symbolic instructions. The print-out takes the following form:

SYMBOLIC INSTRUCTION	DECODED MACHINE EQUIVALENT
SUBTRACT NET	000100000010000000000000100000000
. . .	. . . . .
. . .	. . . . .

When the printing process is complete, the assembly program furnishes the programmer with a group of

punched cards containing the machine language translation so that if the program is used again, the assembly process need not be repeated.

Computer Operations

The format of the instruction word is, for the most part, a precise one. Although slight variations exist, in general the format is as shown in Figure 26.

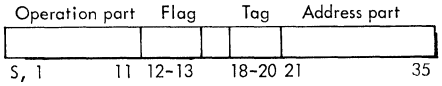


Figure 26. Instruction Format

The operation part usually is contained in positions S, 1-11 of the instruction word. In the following text and in program writing, an alphabetic code is used to identify the instruction operation rather than the full name of the instruction. Thus, the code CLA signifies CLear and Add, or SUB means SUBtract. If the numerical machine language code is to be used, it is given in the octal system. For example, +0500 is the code for the clear and add operation; +0402 denotes subtraction. All abbreviations are simply shorthand methods used to reduce the manual task of writing a program.

The flag part, or flag bits, is contained in positions 12 and 13 of the word. It specifies that indirect addressing is to take place. This operation may be performed only on instructions that use index registers. Both indirect addressing and the use of index registers are explained later in the text.

The tag bits are contained in positions 18, 19, and 20. They specify which index registers are to be used. Only a few instructions may not use indexing.

The address part of the instruction is contained in positions 21 through 35 of the word. It tells the computer the location or "operand" that is to be used with the instruction. In the case of shifting operations, the address part contains the number of places to be shifted. For other instructions, the address part may be a part of the operation itself. Such a case would have the address part expressed as a four-digit number; read-1200 means the reading operation will take place and the 1200 will tell which input-output unit is being used. In this case, the tape unit numbered 0 on data channel A would be used.

In executing the instruction ADD 1000, the computer assumes that the *augend* is in the accumulator and the *addend* (the number being added) is specified by the address part of the ADD instruction. The sign of the numbers (S position) is, of course, considered during an add operation. When two numbers of the same magnitude are being added, the sign of the result is taken from the number in the accumulator.

ACCUMULATOR		STORAGE		RESULT IN ACCUMULATOR
+6	+	−6	=	+0
−6	+	+6	=	−0

The clear-and-add (CLA) instruction is similar to the add instruction except that the accumulator is cleared to zeros and the contents of the location specified by the address part of the CLA instruction are placed in the accumulator.

Add magnitude (ADM) is another similar instruction. Its operation is the same as ADD except that the sign of the number is ignored and the number is treated as positive.

Other arithmetic instructions are treated in a like manner since all arithmetic processes are accomplished by addition, and complementing a result where necessary. Subtraction occurs in the following manner:

7 = 000111	Number in Accumulator
5 = 000101	Number to be Subtracted (Operand)
111000	Complement of Number in Accumulator
+ 000101	
111101	Recomplement the Result (No High-Order Carry)
000010	This is the answer, 5 subtracted from 7 = 2.

Multiplication is accomplished by testing the low-order position of the multiplier and adding the multiplicand if this low-order position is a 1. After each test, the answer is shifted one place to the left. This is repeated until there are no more numbers in the multiplier.

5 = 000101	Multiplicand
×3 = 000011	Multiplier
000101	
000101	
000000	
15 = 00001111	Product

Division is accomplished in a like manner but shifting occurs in the opposite direction. By shifting a binary number one place to the left, the result is the same as multiplying by 2. A number shifted one place to the right has been divided by 2.

A group of word transmission instructions is also provided. These instructions are concerned with the movement, at high speed, of words or parts of words from one location or register to another. In particular, information may be either stored or taken from locations in core storage and various registers in the central processing unit. Since the word transmission instructions are concerned with the movement of data, they are used frequently.

The store (STO) instruction stores the contents of the accumulator in the location specified by the ad-

dress part of the store instruction. Other store instructions accomplish the storing of the address portion, tag, and other portions of both the accumulator and the multiplier-quotient registers.

The computer also has certain load instructions to accomplish the same end as the store instructions, except in reverse. A store instruction places the contents of a register in storage. Conversely a load instruction takes the contents of a storage location and places it in a register. There are also load instructions for the index registers.

With transmit, store, or load type instructions, the contents of the storage location or register from which the data are being moved remain unchanged.

Shift instructions are used to move the contents of the accumulator and/or the multiplier-quotient registers either to the right or the left of their original positions. With the exception of the rotate-MQ-left instruction, zeros are automatically inserted in the vacated positions of the register. Thus, a shift larger than the bit capacity of the register causes the contents of the register to be lost and replaced by zeros. When a shift instruction is interpreted, the amount of the shift is determined by bit positions 28 through 35 of the shift instruction. An example is given in Figure 27.

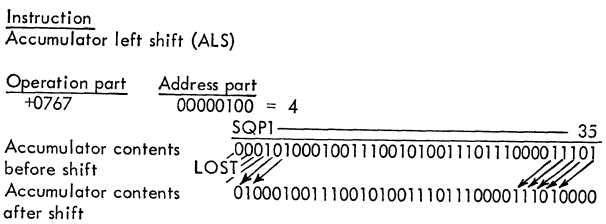


Figure 27. Accumulator Shifting

Control instructions are defined as instructions governing the flow of a program. They are the instructions that cause the computer to alter the normal process of taking its instructions from sequential storage locations. Control or "transfer" instructions may be divided into two types: (1) *unconditional transfers* specifying the location of the next instruction to be executed, and (2) *conditional transfers* performing a test of some kind. The location of the next instruction then depends on the outcome of the test.

Unconditional	Transfer (TRA) 2000.	The next instruction will come from location 2000.
Conditional	Transfer on plus (TPL) 2000.	If the contents of the accumulator are plus, the next instruction will be taken from 2000. If not plus, the next sequential instruction will be taken.

Certain *test* instructions exist and are similar to conditional transfer instructions in that they cause some test to be performed. Unlike conditional transfers, the test instructions do not specify a location for the next instruction. Instead, the alternative location is fixed relative to the position of the test instruction in the program. An example is shown in the following program.

LOCATION	INSTRUCTION	REMARKS
100	Add	
101	P-Bit Test	If the P bit of the accumulator is a 1, the next instruction (102) is skipped and the instruction located after that (103) is executed. If position P contains a 0, the next instruction (102) is executed.
102	Shift Right	
103	Store	

Many of the indicators may be tested in this manner and the flow of the program adjusted to fix the cause of the indication or to proceed around it.

Another important group of instructions is the one concerned with the indexing operations. These instructions are explained under the indexing section of the manual.

Information Paths

The core storage unit is normally connected directly to the central processing unit. It is also the site of the stored program that controls the entire computer system. On the other hand, the auxiliary storage and input-output devices are normally disconnected (not physically but logically) from the system. They become connected only by execution of certain stored program instructions. The contents of these devices may control the computer only after being transmitted to core storage.

Thus, data flows between input-output devices and core storage through the central processing unit in the 704 system (Figure 28).

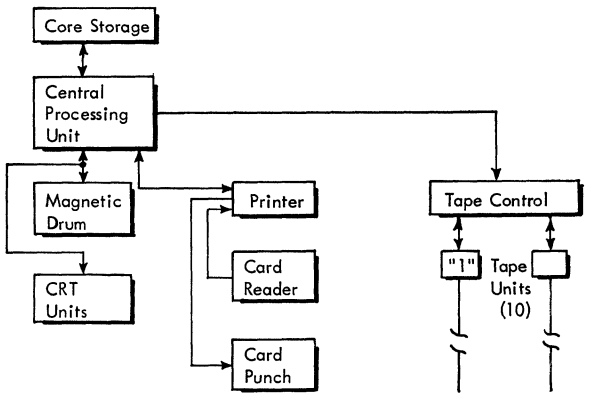


Figure 28. Information Flow, IBM 704 System

Using Figure 28, assume that the magnetic tape unit marked "1" is reading data. Because the tape data are arranged in six-bit groups across the width of the tape, it takes six of these groups to make up a 36-bit word. As these groups are read from tape, they are assembled in the tape control. When a full word is assembled, this word is sent to the central processing unit and from there to the core storage location specified by the instruction that caused the tape to be read. Actually, two instructions are required to read data from any input-output device into core storage. The first is a read-tape-1 instruction with "read" as its operation part and "tape 1" as its address part. The execution of this instruction: (1) selects the proper tape unit, (2) puts it into read status, and (3) starts the tape moving in the proper direction. The second instruction needed is called a copy instruction. "Copy" is the operation part with the location that the data should enter (core storage) as the address part. If a copy-1000 instruction were executed, the data read would be entered into core storage location 1000. A single copy instruction moves *one* full word into storage. If two or more words are to be moved, two or more copy instructions must be furnished, each with a different address part.

INSTRUCTION	ADDRESS	REMARKS
Read	Tape 1	Gets tape 1 ready and moving
Copy	1000	Puts that word into location 1000
Copy	1001	Puts second word into location 1001
Copy	5000	Puts third word into location 5000

This type of instruction group is called a "copy loop" or a copy routine. A true "loop" is fully explained under the heading "Index Registers."

The information paths and the components used in the 709 system are shown in Figure 29. Comparing Figures 28 and 29, note that the major difference between the two systems is the fact that data coming from or going to core storage do not pass through the central processing unit. Instead, a new device called a data synchronizer is used as the go-between for input-output devices and core storage.

With the 709 computer, the stored program starts an input-output operation by defining which device is to transmit (read tape) and what area in storage is to receive the data; then the central processing unit is free to do other calculations. Note the main difference between the 704 and the 709. In the 704, the central processing unit must wait until all data have been transmitted while the 709 merely starts the operation and is then free to do other work. This subject is expanded under the section "IBM 766 Data Synchronizer."

The 7090 computer system uses basically the same information paths as does the 709. Two new units are added in the 7090 system. They replace the data synchronizer's operation. They are called "multiplexor" and "data channel." Their operation is explained in the section "IBM 766 Data Synchronizer." Figure 30 shows information flow and components in the 7090 system.

There may be a maximum of eight data channels attached to the multiplexor. Each data channel may have the same complement of input-output devices as shown in Figure 30. This feature gives a maximum of 80 magnetic tape units, eight card readers, eight card punches, and eight printers.

Figure 30 shows that the central processing unit of the 7090 system has even less control over core storage and the input-output devices than does the 709 system.

The main difference in all three systems is the progression from a synchronous computer (704) to an asynchronous computer (709-7090), as far as the input-output controls are concerned. Here, synchronous is defined as one operation happening after another has finished, while asynchronous means simultaneous occurrence of several operations.

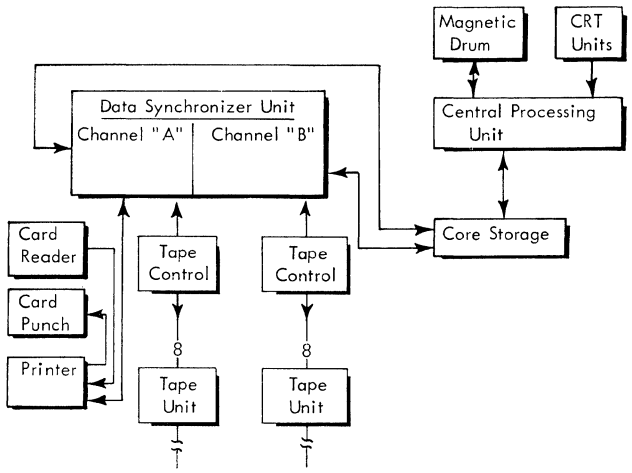


Figure 29. Information Flow, IBM 709 System

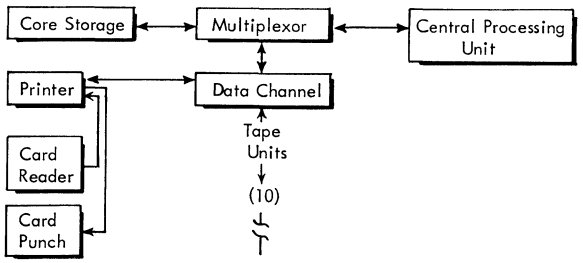


Figure 30. Information Flow, IBM 7090 System

Indexing and Indirect Addressing

The copy routine, described previously, is acceptable if a limited number of words are to be copied. For example, 1,000 copy instructions are needed to copy 1,000 words. This wastes storage space. However, the copy address could be altered by the program each time it is used.

A program to copy 1,000 words from tape number 1 and place them into consecutive storage locations could be as in Figure 31. In this program, thirteen instructions and three constants are required to modify the address of the copy instruction and test for the end of the copy loop. The program is considerably improved over the routine in which 1,000 copy instructions are needed. However, the use of an index register makes the program much more efficient.

The computers contain three index registers that are important to the system's operational abilities. These registers are termed A, B, and C or 1, 2, and 4. The latter terminology is more convenient for the programmer because the numbers used are the octal representation of the "addresses" of the three index registers.

Inst. Loc.	Instruction	Remarks
50	Read tape 1	Get tape 1 ready to read
51	Copy 200	Put the 1st word into location 200
52	CLA 62	Clear the accumulator to zeros and add to it the contents of location 62 (0000----000)
53	ADD 64	Add to the accumulator contents the contents of location 64 (0000----001). These instructions make a counter which is increased by one for each copy instruction executed.
54	STO 62	Store the result in location 62.
55	CLA 63	Clear and add into the accumulator, the contents of location 63 (0000----1000).
56	SUB 62	Subtract from the accumulator (1000) the contents of location 62. (This is the location that will increase with each copy instruction execution.)
57	TRZ 65	Transfer when the accumulator contents are zero. This will occur when 1000 copies have been executed.
58	CLA 51	Clear the accumulator and add the contents of location 51 (copy 200).
59	ADD 64	Add the contents of location 64 to the contents of the accumulator. This will increase the copy address from 200 to 201 the first time. The address will then be increased by one each time the instruction is executed.
60	STO 51	Store the result back at location 51.
61	TR 51	Transfer back to location 51 so that the next word will will be read in. (Remember that the copy address has been increased by one.)
62	00---0000	Constants:
63	00---1000	
64	00---0001	
65	STOP	The program will reach this instruction after 1000 words have been copied and will stop.

Figure 31. Sample Program

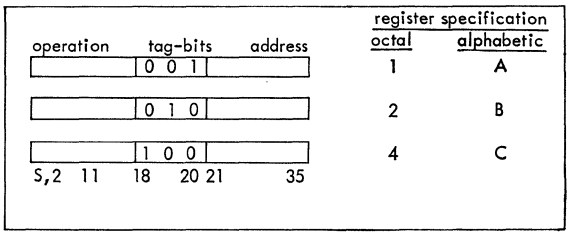


Figure 32. Index-Register Tag Bits

The addresses of index registers are stipulated in a part of the instruction word known as the *tag field.* Such addresses are normally referred to as *tags.* The tags perform the same function as the address of the instruction. They tell the computer whether an index register is to be used and which register is to be used. The tag field is located in positions 18, 19, and 20 of the instruction (Figure 32). By having more than one tag bit in the tag field, two or more index registers may be used in a single instruction. Thus, the contents of the registers would be combined and the resultant sum would be used.

The index registers are 15 positions long—large enough to hold the largest possible storage address. They are used to modify an address by adding the complement of their contents to the address. This reduces the address by the contents of the index register. Many instructions may specify index action, thus making them useful for such functions as address modification and counting.

As an example of the arithmetic involved when index registers are used, assume that index register 1 contains the number 2 and that the copy instruction, with an address of 200, is to be executed. The following occurs (Figure 33).

When the copy instruction is decoded, the tag bit in position 20 specifies index action; thus, the contents of index register 1 are complemented and placed in the adders. Note that the contents of an index register are *always* complemented when sent to the adders. This feature results in subtracting the contents from the address. The address portion of the copy instruction is also placed in the adders; after adding the two numbers, the result (called the *effective address*) is used in execution of the copy instruction instead of

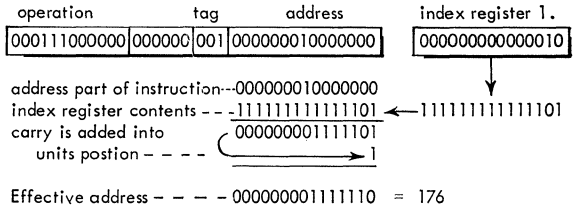


Figure 33. Index-Register Arithmetic, Subtracting

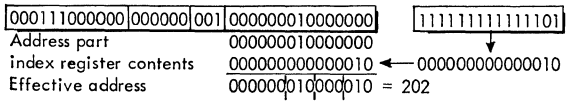


Figure 34. Index-Register Arithmetic, Adding

the actual address. In this case, the effective address is 176. If the programmer wishes to increase the effective address, the number to be placed in the index register is inserted in complement form. Thus, when the address and index register contents are combined, the result is an additive process. Using the same facts (as in Figure 33) with the index contents in complement form, the effective address is now 202 instead of 176 (Figure 34).

With an instruction available to reduce the contents of the index register each time it is used, the effective address is easily modified. The instruction is called "transfer on index" (TIX) and has the format shown in Figure 35. The operation of the instruction is as follows: If the number in the specified index register is greater than the decrement portion, the index register is reduced by the amount of the decrement and the computer takes its next instruction from the location specified by the instruction address. If the index register contents are equal to or less than the decrement, the register is not changed and the next instruction in sequence is executed.

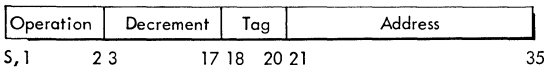


Figure 35. TIX Instruction Format

To apply this instruction, assume the same conditions set forth for Figure 31. Follow the copy instruction with the transfer-on-index instruction with a decrement portion of 1. The result would be as follows:

True Index Register Contents	000000000000010
Decrement Contents of TIX Instruction	000000000000001
True Result after Decrement Is Subtracted	000000000000001

This use of index registers for address modification can now be applied to a program to copy 1,000 words from a tape unit into 1,000 storage locations. The program can be written as follows:

LOCATION	INSTRUCTION DECREMENT	TAG	ADDRESS
50	Read tape		1
51	Load index register	1	100
52	Copy	1	1200
53	Transfer on index (0001)	1	52
54	Compute		
.	. . .		
.	. . .		
100	Constant (1000)		

Location 50. This instruction selects tape unit 1 for reading.

Location 51. The load index instruction places the contents of location 100 into index register 1. Location 100 contains the quantity 1000.

Location 52. Because the instruction is tagged, the address of the copy instruction is treated as explained in Figure 33. The first time CPY is executed, the contents of the index register are subtracted, resulting in an effective address of 200. The index register is then reduced by the decrement of the transfer-on-index instruction so that successively higher locations of storage are used each time through the loop.

Location 53. The TIX instruction, having a decrement of 1, reduces the index register each time it is executed for a total of 1,000 times. After 1,000 words have been copied, the decrement of the TIX will equal the index register. No transfer will occur and the next sequential instruction (54) will be executed.

Location 100. The constant of 1,000 is needed to load the index register when the instruction at location 51 is executed.

By comparing this routine, properly called a copy loop, with the routine explained before, the number of instructions needed to copy 1,000 words from tape has been reduced from 1,001 to four instructions and one constant (through the computer's ability to count, modify addresses, and test for the end of loop with one instruction).

By using the many indexing operations available in the computer, an index register may be added to, subtracted from, and tested in various combinations so that many desired effects may be accomplished.

If more than one index register is specified by the tag field, the procedure is the same except that the contents of two or more index registers are combined and the result is then subtracted from the address field of the instruction being executed.

The variable address concept is expanded and includes a feature called *indirect addressing.* Just as index registers are addressed by tags, indirect addressing is specified by the presence of 1's in positions 12 and 13 of the instruction. This portion is called a *flag.* For example, assume that the instruction shown (Figure 36) is to be executed. Instead of subtracting the contents of location 2054 from the contents of the accumulator register, the computer, because of the

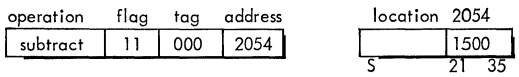


Figure 36. Instruction Format

flag, examines the location specified by the address (2054) and uses the address part of that location (1500) to obtain an effective address. The subtraction is then executed as if its address part had contained 1500 instead of 2054. Thus, the contents of location 1500 are subtracted from the contents of the accumulator register.

The indirect addressing feature may be combined with indexing to obtain even greater flexibility in operation. Using the same example and adding an index register tag, the operation would be as shown in Figure 37.

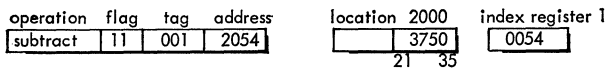


Figure 37. Indirect Addressing and Indexing Format

The contents of index register 1 are subtracted from the address portion of the subtract instruction giving an effective address of 2000. Location 2000 is then examined and its address portion (3750) becomes a second effective address. Thus the contents of location 3750 instead of 2054 are subtracted from the contents of the accumulator.

Sample Problems

The following figures show a few examples of some of the problems that computers are solving today. Although the examples shown are in scientific fields, the computers are adaptable to business applications. They are used for payroll, material inventory, control of crude oil processing, and many other business applications where efficient operation depends upon prompt, high-speed results.

The first application shown is in the field of scientific research (Figure 38).

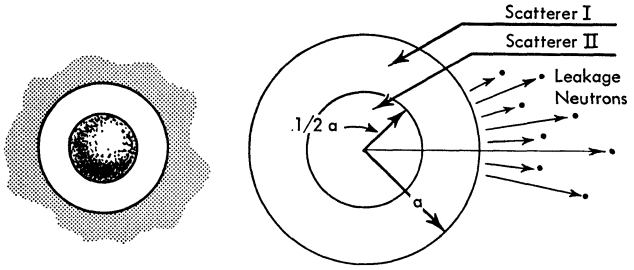


Figure 38. Neutron Scattering

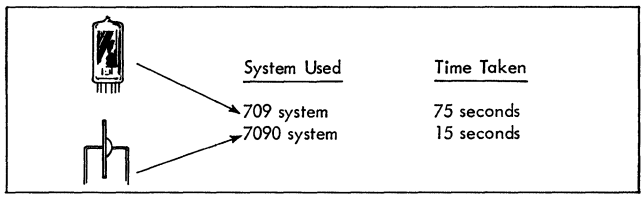


Figure 39. Time Comparisons, Neutron Scattering Problem

Neutron Scattering

Given:

1. A spherically symmetric system consisting of two scattering materials.
2. An initial neutron distribution in space and energy.
3. Scattering laws for the materials and dimensions for the system.

Find:

1. The neutron leakage rate from the system, caused by scattering.
2. The asymptotic neutron distribution in space and energy.

The initial collision distributions are combined according to the scattering laws and are inserted into matrix equations. These are multiplied by the matrices and a new set of distributions is obtained. The procedure is repeated on an iterative basis until the asymptotic or limiting phase is reached.

Time comparisons for the solving of 25 of these iterations appear in Figure 39 and are approximate.

Flight Trajectory

The second application is in the field of national defense (Figure 40). The path of an object which is launched from some arbitrary point is calculated. Its trajectory is determined by its flight characteristics,

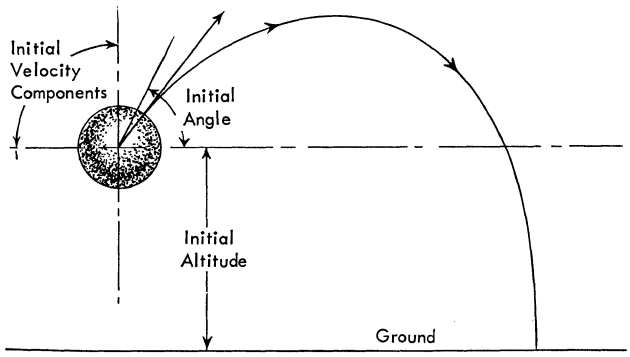


Figure 40. Flight Trajectory

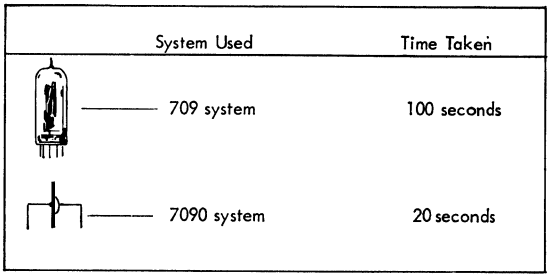


Figure 41. Time Comparison for Each Trajectory

the initial components of its velocity, an initial altitude, and the initial angle.

The position of the object is identified by its coordinates with respect to fixed axes passing through the launching point.

Factors in this problem are:

1,100 time steps per trajectory.
1,000 operations per time step.
1,100,000 operations per trajectory.

The time comparisons for each computer are approximate and are shown in Figure 41.

Electron Density

The third application is in the field of fundamental research (Figure 42). The electrons surrounding two nitrogen atoms which form a nitrogen molecule) are treated as a degenerate Fermi-Dirac gas. Such treatment results in a partial differential equation called the Thomas-Fermi-Dirac or "statistical field" equation.

The computer calculates solutions to this equation and thereby determines the density of electrons in the molecule and the energy of the molecule. The Thomas-Fermi-Dirac equation is replaced by a difference system over a network of points. The network consists of the intersections of 60 quasi-lemniscates

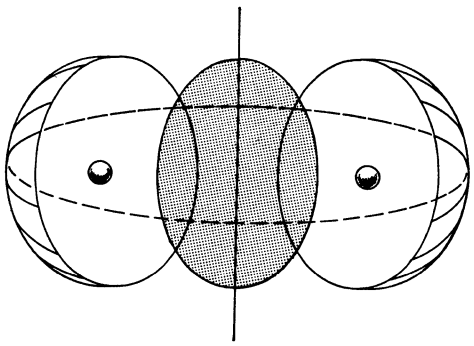


Figure 42. Electron Density

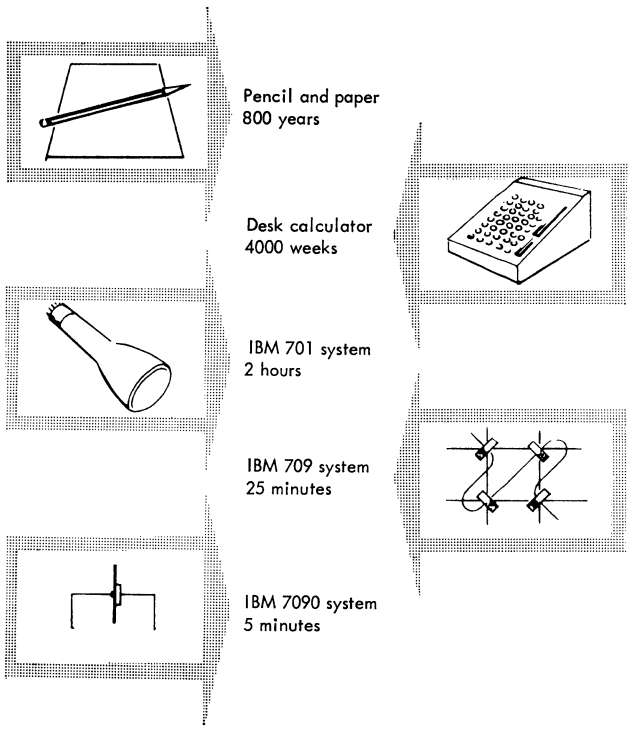


Figure 43. Time Comparison of Computer Systems, Electron Density Problem

with 15 curves at right angles to them. The following procedures are used to solve the problem:

900 simultaneous non-linear equations.
80 iterations per solution.
900,000 operations per iteration.
72,000,000 operations per solution.

This application graphically points out one of the problems that could not be solved without the use of high-speed computers. Again, the time comparisons are approximate. Note the great range of times. The span of 800 years required with a pencil and paper is reduced to five minutes by the IBM 7090 Data Processing System (Figure 43).

Average execution speeds for both fixed and floating-point arithmetic operations (including the time needed to take a word from or put a word into core storage) is shown in Figure 44.

The use of transistors in the 7090 system provides a saving in electrical power and air conditioning requirements. The reduction of electrical power may be as great as seventy percent of the 709 requirements.

The modular unit design of the 7090 makes it possible to place units side by side and thus minimize space requirements. As much as fifty percent of the 709 system space requirements may be saved.

FIXED POINT OPERATIONS				
Operation	709 Time and Op/Sec		7090 Time and Op/Sec	
Add or Subtract	24 usec*	41,660	4.4 usec	227,270
Logical Operations	24 usec	41,660	4.4 usec	227,270
Multiply	190 usec	5,260	25.3 usec	39,640
Divide	240 usec	4,160	30.5 usec	32,780
FLOATING POINT OPERATIONS				
Operation	709 Time and Op/Sec		7090 Time and Op/Sec	
Add or Subtract	84 usec	11,900	14.0 usec	71,420
Multiply	170 usec	5,880	24.0 usec	41,660
Divide	216 usec	4,620	28.3 usec	35,330
Operations-per-second numbers are approximate.				
*usec = microseconds				

Figure 44. Average Execution Speeds

Operator's Consoles

The 709 operator's console is a physical part of the central processing unit (Figure 45). It consists of neon lights, incandescent lamps, keys, and switches. The contents of the main central processing unit registers are displayed in neon lights. A 1 is represented by the light's being on; a 0, by the light's being off. The 36 entry keys are available to manually insert a word or words into the central processing unit and core storage.

The IBM 7151 Console Control (Figure 46A) is a separate unit providing centralized control of the 7090 system. It contains indicators, switches, keys, and register displays for the operator's use. Channel indica-

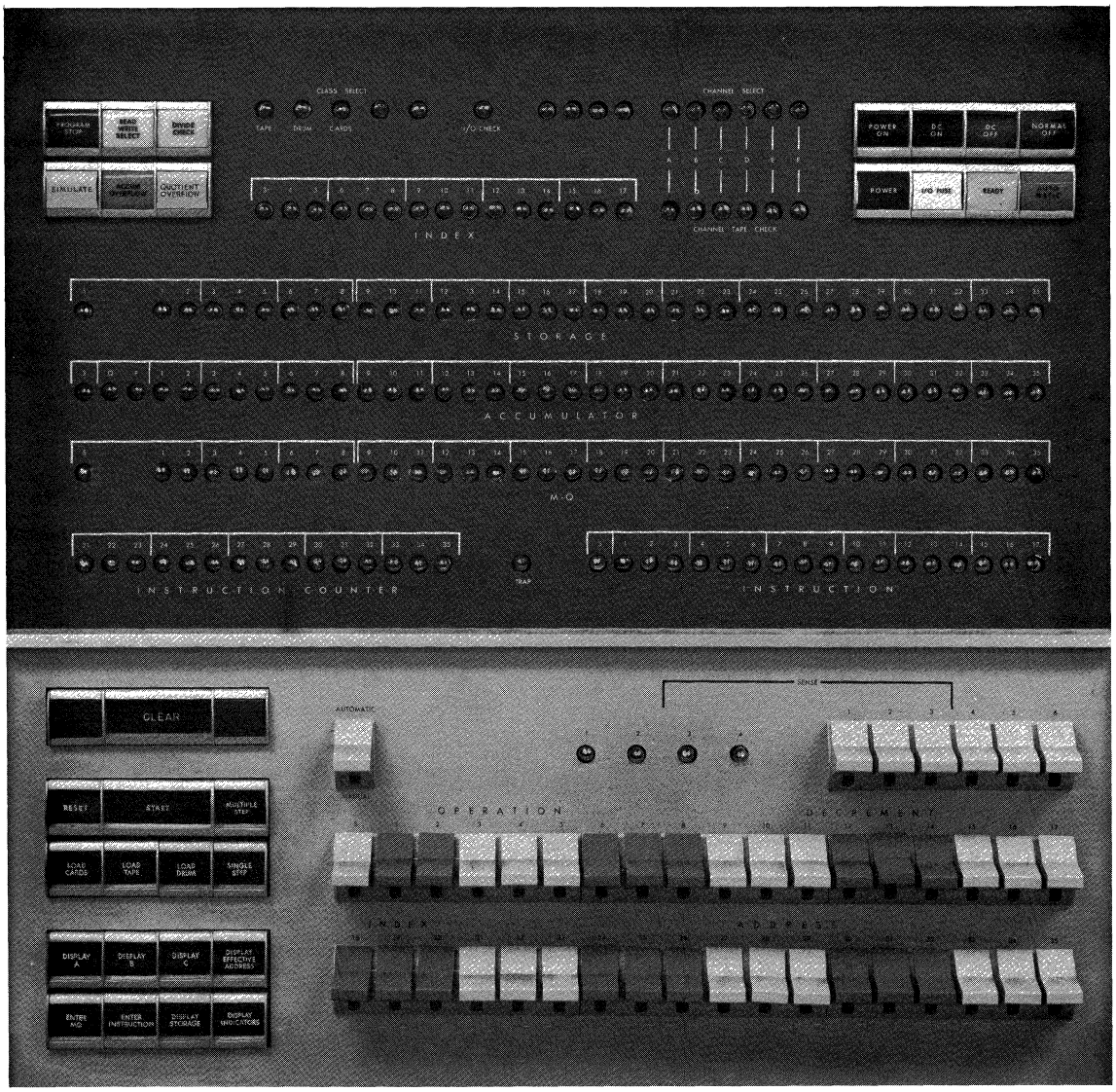


Figure 45. IBM 709 Console

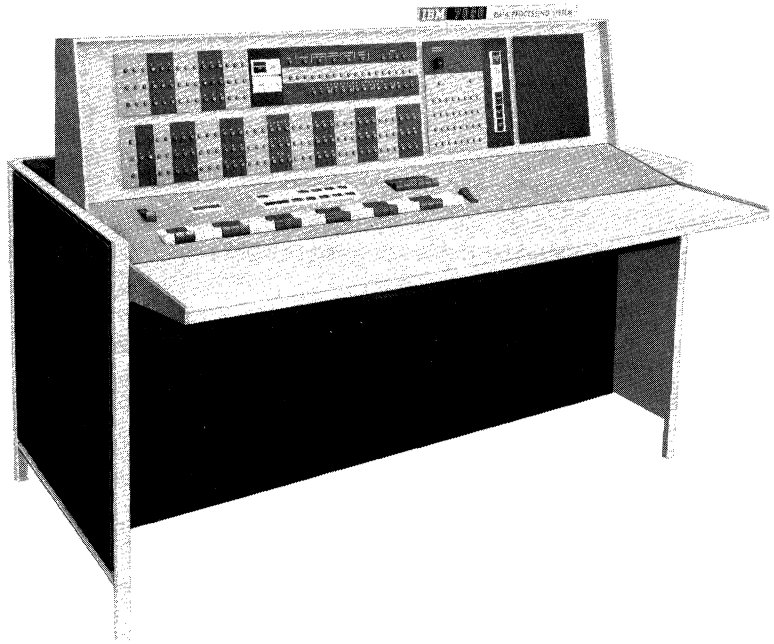


Figure 46A. IBM 7151 Console Control (7090)

tors for data channel operation are provided and the register displays have been grouped for convenience. Data in any storage location can be displayed and/or changed by manual insertion through use of the entry keys and switches.

A maintenance feature, marginal checking device, is also a part of the console. This feature includes the capability of varying voltages and frequencies during programmed diagnostic testing to detect potential difficulties before actual failure of components occurs.

The IBM 7617 Data Channel Console (Figure 46B) provides for greater input-output flexibility and efficiency between the IBM 7090 Data Processing System and its operator. One data channel console is used with each IBM 7607 I or II Data Channel. The console may be located up to 50 feet away from the data channel, permitting the computer system console and data channel consoles to be grouped for greater operating efficiency.

Several operations may be performed from the data channel console when the console is in a manual mode. These include:

1. Data transmission operations
2. Non-data tape operations
3. Loading operations
4. Storing operations

Figure 46B. IBM 7617 Data Channel Console

Magnetic Tape Storage

The principal input-output medium for the computer is magnetic tape. The tape is used, in addition to input-output functions, for storage of intermediate results. It is also used as permanent storage for large files of data. Magnetic tapes may be re-used many times, because old data are automatically erased as new data are recorded.

The tape is a plastic material that is coated on one side with a metallic oxide. It is one-half inch wide and is packaged on reels with tape lengths as great as 2,400 feet. The tape can be easily magnetized and yet retains the magnetized "spots" when put in static storage for indefinite periods of time.

The basic electronic principle of magnetic tape recording, as used in data processing systems, is similar to that of a home tape recorder. Instead of recording music or voice, business and scientific data are recorded in the form of magnetized spots. A schematic of the recording head together with a section of magnetic tape is shown in Figure 47.

When electrical current flows through the recording head coil, the magnetic oxide particles on the tape are oriented in one direction. If the current in the coil reverses its direction, the particles on the tape will be oriented in the opposite direction. This is called reversing the polarity of the magnetic flux. Each such change of polarity is given the value of 1. If no change occurs, the value is given as 0.

Now, if the tape is moved past the recording head and the current in the coil is alternated at time intervals, writing on magnetic tape is accomplished.

To read the tape, the process is reversed. As the magnetized "spots" pass the recording head, small voltages appear in the recording coil. These voltages are amplified (as in core storage) and are then used by the computer.

The advantages of tape storage are shown in Figure 48.

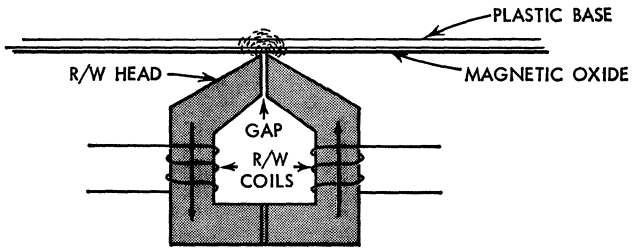


Figure 47. Magnetic Tape Recording

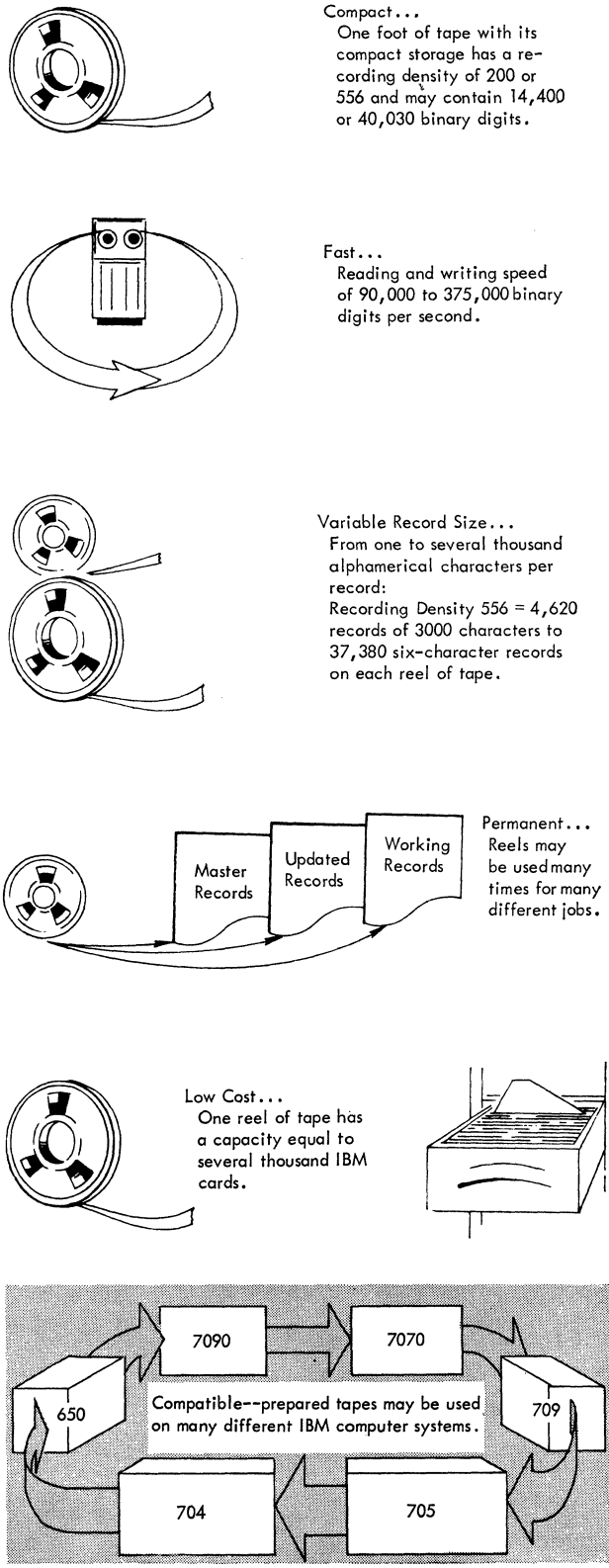


Figure 48. Magnetic Tape Advantages

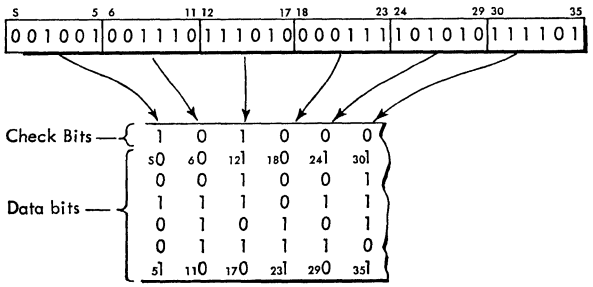


Figure 49. Binary Word Recorded on Tape

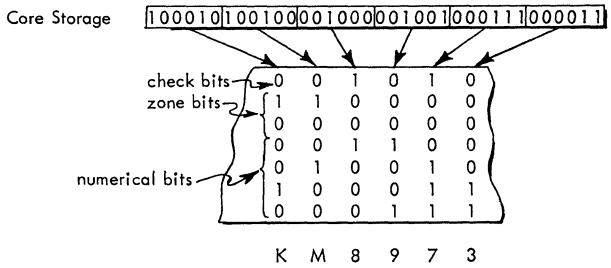


Figure 50. BCD Word Recorded on Tape

During actual tape unit operation, seven recording heads instead of one are placed vertically across the tape. Thus, recording occurs in seven columns or "tracks." With the 709 or 7090 systems, two different "modes" of recording are used, binary and binary coded decimal (BCD).

If data are recorded on magnetic tape just as they are found in core storage (except for the check bits), the tape is said to be a binary tape or to have been recorded in the *binary mode.* The seven recording tracks are used as follows:

1. One track, C, is used for check-bit recording.
2. Six tracks, B, A, 8, 4, 2, and 1, are used to record six bits of the 36-bit word.

Thus, a binary word requires six recordings of six bits each, as is shown in Figure 49.

The check track contains a 1 bit whenever the vertical (across the tape) sum of the 1's in the B, A, 8, 4, 2, and 1 tracks is even. This check bit is computed by the tape control in the 709 and the data channel in the 7090.

Alphabetic and decimal information may be recorded on, or read from, a magnetic tape without depending upon the computer through use of auxiliary equipment.

Magnetic tapes prepared or used on this equipment have a special coding system known as *binary-coded-decimal* (BCD). As the six-bit BCD characters are read from the tape, some of the characters are altered. This alteration is performed so that the digits 0 through 9 and the characters A through Z are represented in core storage by six-bit binary numbers. The alteration of characters is shown in the following table:

CLASS	IN CORE STORAGE		ON TAPE	
	B	A	B	A
Numerical	0	0	0	0
A to I	0	1	1	1
J to R	1	0	1	0
S to Z	1	1	0	1

The digits 1 through 9 are represented by the six-bit binary numbers 000001 through 001001 (their exact values as binary integers). Thus, the B and A (zone parts) of these digits would be 00. The number zero is represented on tape by the bit configuration 001010. This representation would be automatically altered to 000000 when reading in the BCD mode.

During writing in the BCD mode, the alteration procedure is reversed so that the BCD characters in storage are transformed to the BCD tape format. The zone portions (B and A bits) are altered and the number zero (000000) is replaced by 001010.

In addition to alphabetic and numerical characters, the BCD format provides for punctuation marks and other special symbols. Included is the BCD character BLANK which suppresses printing or punching in any desired position during auxiliary operations.

Figure 50 shows the characters K M 8 9 7 3 as they would appear on tape and in magnetic core storage. Each word in core storage may contain six BCD characters. Note that the check bit occurs when the sum of the 1's in both the numerical and zone tracks is *odd.* Tapes prepared in the BCD mode are compatible with those prepared on the IBM 650, 702, 705, and 7070 Data Processing Systems. Figure 51 represents a section of magnetic tape with all of the possible characters recorded on it in the BCD mode.

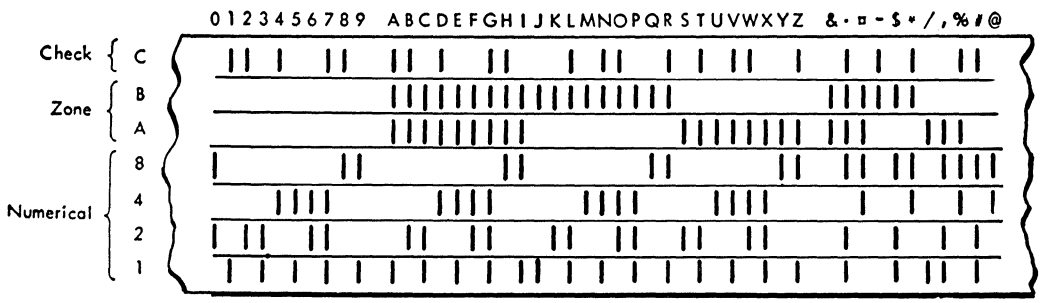


Figure 51. Tape Character Coding.

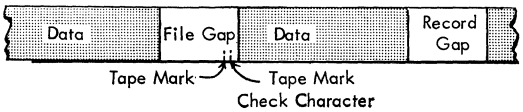


Figure 52. Record and File Gaps on Magnetic Tape

As information is recorded on tape, the number of 1's in each track across the tape is automatically counted and a check bit (1) is placed in the seventh, or C, track. When the tape is read, the check bit is recomputed and compared against the recorded check bit. Any discrepancy will signal an error in tape operation.

In addition to this vertical checking of each column, a horizontal count of each track containing 1's is taken and recorded at the end of the record as a separate column called "longitudinal redundancy check character" (LRCR).

A word or group of words, recorded consecutively on tape, is referred to as a *record.* The number of words making up a record is unrestricted within the storage limitations of the system itself. If more than one record is placed on the same tape, a ¾ inch section of blank tape called a *record gap* is placed between them. Records may also be grouped together forming *files.* These files are separated by other blank sections of tape called *file gaps.*

This file gap consists of a section of blank tape and includes a special character called a *tape mark.* This character and/or gap defines the difference between the records and files on the tape (Figure 52). The number of words in a record and the number of records in a file are variable and are determined by the stored program and the limitations of the system.

When data are to be recorded on tape, a full word of 36 bits is taken from storage and is subdivided into groups of six bits each. Each group, with its associated check bit, is recorded in one column across the tape. Tapes are written or read in the forward direction only. However, the same tape may be written, backspaced, and then read or rewritten if desired. Backspacing is halted by the record gap or file gap; thus, one backspace instruction results in moving the tape in a backward direction to the next record or file gap.

Because the writing operation automatically erases any previous information recorded on the tape, a *file protection device* is provided to prevent accidental erasure of information. A circular groove is molded around the center of each tape reel to fit a demountable plastic ring. Without the ring in place, writing is suspended and only reading may occur. The reel in this condition is protected. When the ring is in place, either writing or reading may be performed (Figure 53).

Like movie film, the tape must have a short length of blank space at the beginning and at the end of the reel that 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 tape, are photoelectrically sensed to indicate the physical end of tape and the starting point for recording (Figure 54). During writing operation, the reflective spot signals that the end of the reel has been sensed.

Figure 55 shows schematically the position of the tape reels in relation to the read-write head, feed rollers, and the vacuum columns. While reading or writing, tape is transported from the file reel (left side) past the recording head to the machine reel (right side).

Since it is impossible to start and stop high-speed motion of the tape without some slack in the tape, a

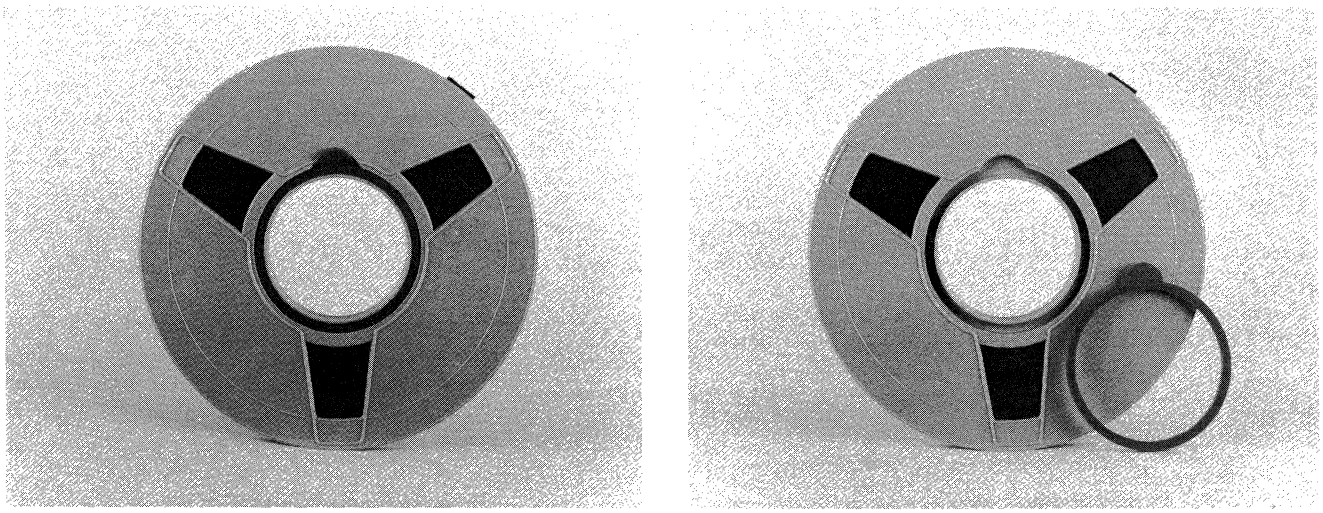


Figure 53. File Protect Ring

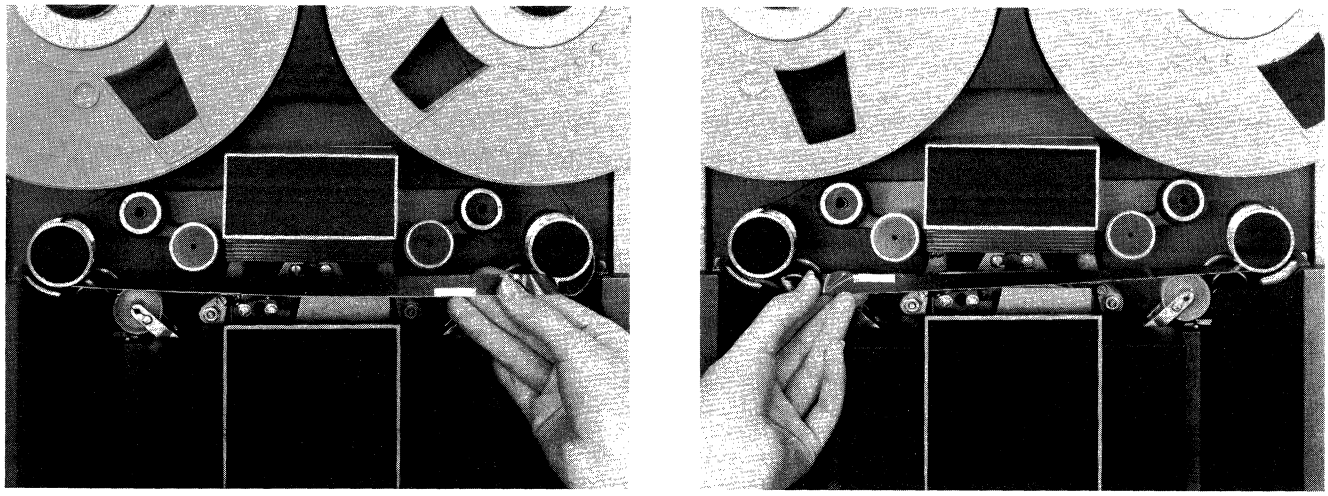


Figure 54. Reflective Spots on Tape

loop of tape is held in the vacuum columns and acts as a buffer for this motion. As tape is drawn from one column, it is replenished from the reel above it. As it is fed into the opposite column, the associated reel takes up the slack.

The read-write head assembly consists of two parts, the lower of which is stationary, and the upper part moves up and down under control of the load and unload keys. Tape threading must be done with the tape unit in an unload status. (Changing reels or threading tape takes approximately two minutes.)

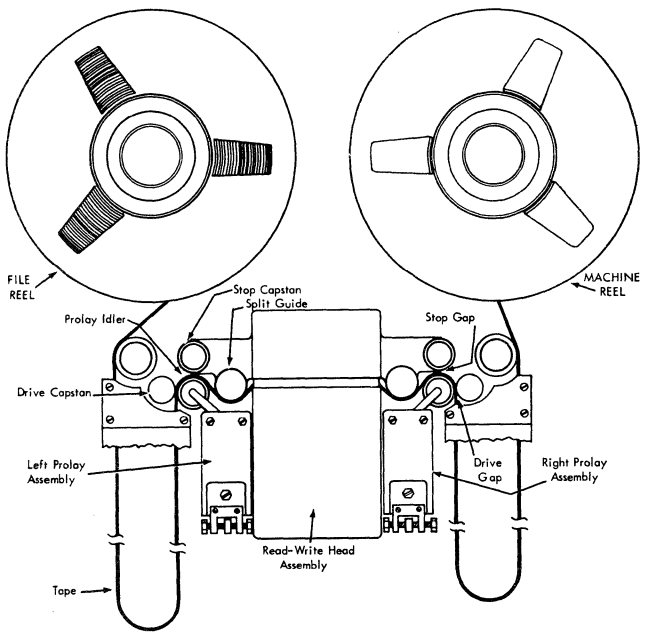


Figure 55. Schematic, Tape Feed

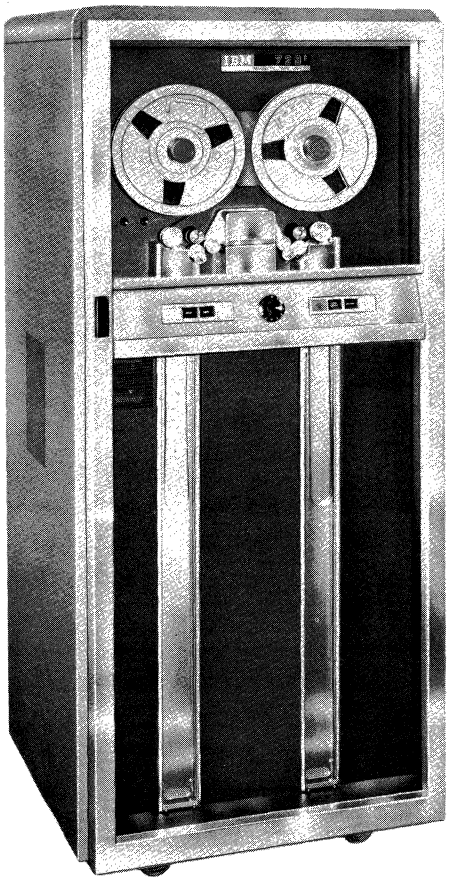
Rewind is under the control of a stored program instruction or may be initiated by depressing the load-rewind key on the unit. If more than one-half inch of tape has been wound on the machine reel, the unit automatically performs a high-speed rewind (500 inches per second). The head is raised, tape is pulled from the vacuum columns and the tape is moved in a reverse direction until less than one-half inch of tape remains on the machine reel. When this occurs, the tape is lowered into the columns, the head assembly is lowered and the backward movement is reduced to the normal operating speed of the unit.

During the high-speed rewind, tape is passed between a light source and a photo cell. If the tape breaks, the light strikes the photo cell causing tape motion to immediately stop.

The IBM 729 I Magnetic Tape Unit is used with the 709 systems; 729 II and 729 IV units are used with the 7090 system (Figure 56).

Each unit is equipped with a selector dial to be set from 0 through 9 by the operator. The selected number then becomes the "address" of the unit. Instructions from the stored program condition or alert the unit for use by specifying this address.

The 729 I reads and writes at a density of 200 characters per inch of tape. This means that a 100-character record would occupy one-half inch of tape. Tape is moved past the read-write head at a speed of 75 inches per second, thereby making the rate of reading or writing 15,000 characters per second or 67 microseconds per character.



IBM 729 I

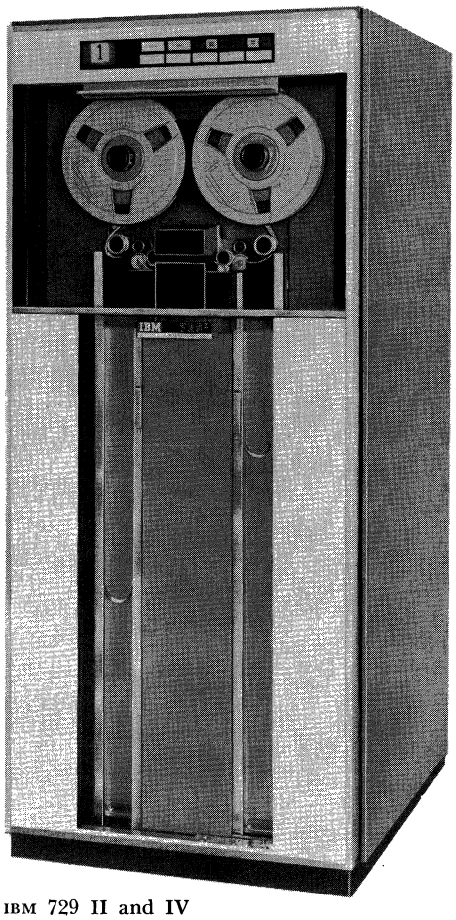
Figure 56. IBM Magnetic Tape Units

The 729 II unit reads and writes at one of two character densities, either 200 or 556 characters per inch of tape. This dual density feature is under stored program control. Tape moves past the read-write head at a speed of 75 inches per second. The combination of character density and tape speed give character rates of 15,000 or 41,667 characters per second depending upon the recording density used.

Since a record gap is placed between each record on tape, the total time required to read a record must include time to space over the gap. This is termed "access time" to the data. Average access time for the 729 I and 729 II is 10.8 milliseconds.

Figure 57 is a table showing a comparison between 100 different size records written in both density modes on the 729 II tape unit. Note that 2,370 feet of the available 2,400 feet are used, allowing 30 feet for the beginning- and end-of-tape reflective strips.

The 729 IV tape unit also reads and writes at two densities, 200 or 556, and is under program control.



IBM 729 II and IV

With this model, however, the tape moves at a speed of 112.5 inches per second. Thus, character density and tape speed give character rates of 22,500 and 62,500 characters per second. For the 729 IV, with its faster tape speed, the average access time is reduced to 7.3 milliseconds. Figure 58 shows a time comparison for 100 records on the 729 IV tape unit.

Figures 57 and 58 show that the effective character rate increases in efficiency as longer records are processed. Rate per character for the 729 II is 67 or 24 microseconds; for the 729 IV, it is 44 or 16 microseconds. Figure 59 shows rates for 100 records.

The 729 Magnetic Tape Units incorporate two magnetic gaps for each of the seven recording tracks. One

Characters Per Record	729 II - Low Density Time in Seconds	729 II - High Density Time in Seconds
6	1.12	1.09
36	1.32	1.17
360	3.49	1.94
720	5.90	2.81
1800	13.14	5.40
3600	25.20	9.72

Figure 57. Time Comparison for the IBM 729 II

Characters Per Record	729 IV - Low Density Time in Seconds	729 IV - High Density Time in Seconds
6	.76	.74
36	.89	.78
360	2.31	1.30
720	3.90	1.88
1800	8.65	3.61
3600	16.57	6.49

Figure 58. Time Comparison for the IBM 729 IV

Characters Per Record	729 I, 729 II - Low Density Characters per Second	729 II - High Density Characters per Second
6	536	550
36	2727	3077
360	10315	18556
720	12203	25623
1800	13699	33333
3600	14286	37037
	729 IV - Low Density	729 IV - High Density
6	789	811
36	4045	4615
360	15584	27692
720	18461	38298
1800	20809	49861
3600	21726	55470

Figure 59. Effective Character Rates for 100 Records

gap is used for writing; the other, for reading. The two-gap head (Figure 60) offers increased checking while writing. A tape that is being written passes first over the write gap (to record the data) and then over the read gap (to check the writing). With this type of operation, written data are automatically read and

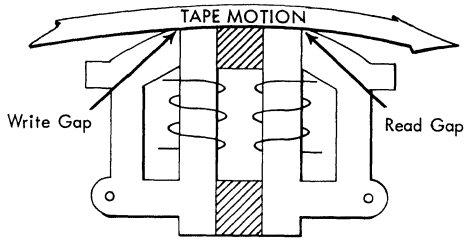


Figure 60. Two-Gap Recording Head

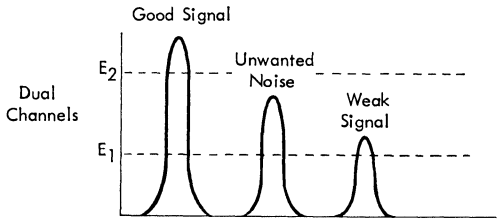


Figure 61. Error Detection

checked. Each vertical column of six bits and the check bit for that column are analyzed. If any discrepancy occurred during the write operation, it is detected at the read gap and a tape check indicator is turned on.

Dual-level sensing is used with the two-gap head, to increase error or weak signal detection at the time of writing. Signals received by the checking circuitry associated with the read-write head are interpreted at two different energy levels (E1 and E2 in Figure 61). This interpretation or analysis determines that:

1. The data signal strength is at a level that provides good signals when the tape is read at a later time.
2. No unwanted signal (strong noise) is present on the recorded tape.
3. Compensating errors do not exist because of a bit-for-bit match that occurs at both levels.

By testing the tape check indicator, the computer can institute corrective action whenever an error is sensed.

Figure 62 shows the compatible features of the different magnetic tape units and data processing systems.

		727	729 I	729 II	729 III	729 IV
Tape Speed (Inches per Second)		75	75	75	112.5	112.5
Record Density (per Inch)		200	200	200 or 556	556	200 or 556
Character Rate per Second		15000	15000	15000 or 41667	62500	22500 or 62500
Record Gap Size 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 Control Unit:	IBM 753	10				
	IBM 754	10				
	IBM 755		8			
	IBM 760	2				
	IBM 767(1)		5	and/or	5	
	IBM 777	8				
	IBM 7607(2)			10	or	10
Where used:	Aux. Card-to-Tape (3)	x	x			
	Aux. Tape-to-Card	x	x			
	Aux. Tape-to-Printer					
	IBM 717	x	x			
	IBM 720-730	x	x			
	IBM 650	x				
	IBM 704	x				
	IBM 705 I or II	x				
	IBM 705 III		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 62. Features of Magnetic Tape Units

Auxiliary Equipment

Auxiliary equipment, a part of most data processing systems, is described below.

1. Card-to-Tape (Figure 63)

The IBM 714 Card Reader, IBM 727 or 729 I Magnetic Tape Unit, and the IBM 759 Card Reader Control perform independent card-to-tape operations. Information can be read from a punched card and written on magnetic tape for future processing in the computer at a speed of 250 cards per minute. Auxiliary card-to-tape equipment does the following:

a. It converts punched card data into magnetic tape records without using the computer.
b. It selects or rearranges the data on cards for recording.
c. It emits constant information.
d. It checks each record automatically after it is recorded.
e. It selects data from two cards, if desired, to be combined into one record on tape.

2. Tape-to-Card (Figure 64)

The IBM 722 Card Punch, IBM 727 or 729 I Magnetic Tape Unit, and the IBM 758 Card Punch Control perform independent tape-to-card operations. Information can be read directly from magnetic tape and punched into cards. The tape can be recorded either on card-to-tape equipment or on one of the IBM data processing systems. Tape to card equipment performs the following operations:

a. It converts magnetic tape records to punched cards independent of computer operations.
b. It automatically checks punched data for accuracy.
c. It operates the card punch at a speed of 100 cards per minute.

3. Tape-to-Printer (Figure 65)

The IBM 717 Printer, IBM 727 or 729 I Magnetic Tape Unit and the IBM 757 Printer Control perform independent tape-to-printer operations. Information can be read from tape and printed in the exact form in which it appears on tape. The tape can be recorded either on card-to-tape equipment or on one of the IBM data processing systems. The equipment performs the following:

a. It converts magnetic tape records to printed forms without depending on computer operations.
b. It prints 150 lines per minute, using 120 type wheels of 48 characters each. Alphabetic, numerical, and 11 special symbols are used.
c. It checks printed information automatically.

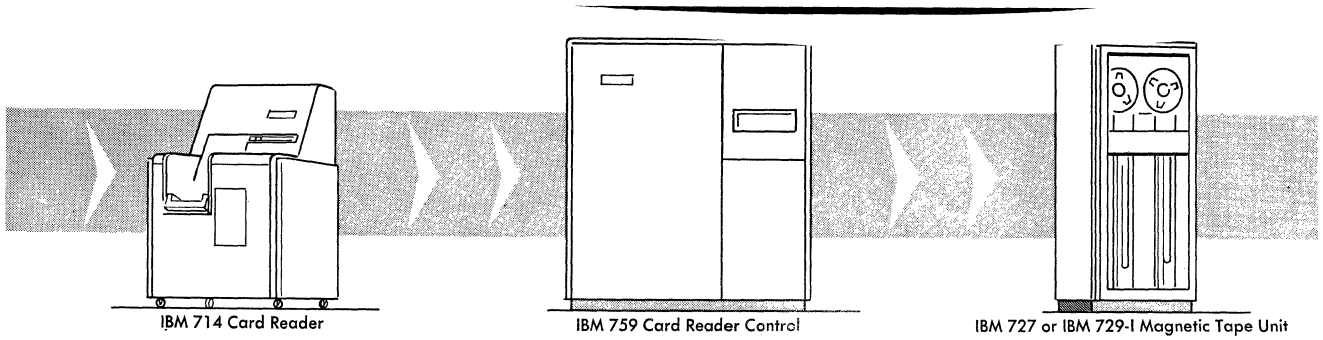


Figure 63. Card-to-Tape Auxiliary Units

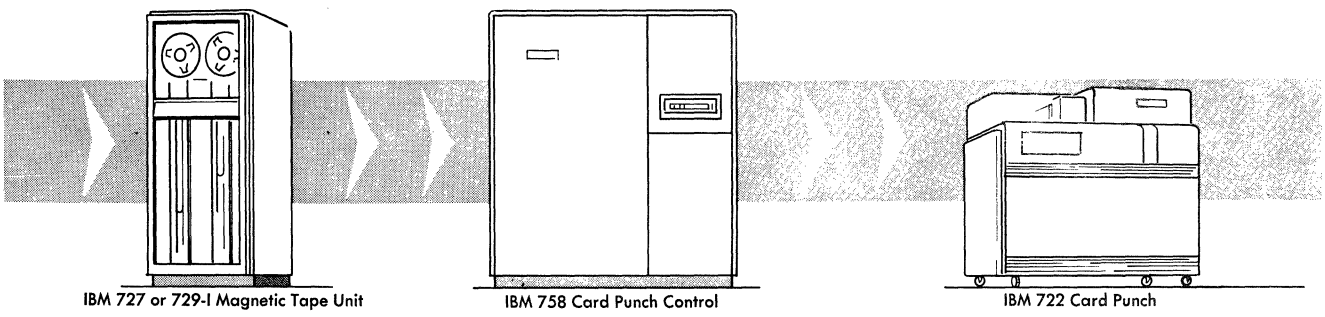


Figure 64. Tape-to-Card Equipment

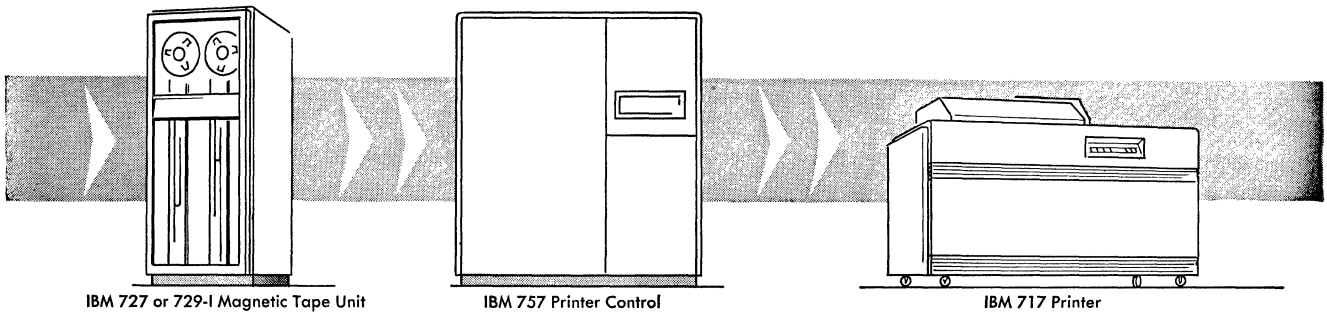


Figure 65. Tape-to-717 Printer

4. *Tape-to-High Speed Printer (Figure 66)*

The IBM 720 or 730 Printer (Model 2), IBM 727 or 729 I Magnetic Tape Unit, and the IBM 760 Control and Storage may perform independent tape-to-printer operations. Information can be read from magnetic tape and printed in the form it appeared on tape. The tape can be recorded either on card-to-tape equipment or on one of the IBM data processing systems. Auxiliary high-speed printing from tape does the following:

a. It converts magnetic tape records to printed form without depending upon computer operations.
b. It prints 500 lines per minute with the 720, or 1,000 lines per minute with the 730 Printer.
c. It uses 120 printing positions of 47 characters each. Alphabetic, numerical, and special symbols can be printed.
d. It checks information automatically.

5. *Auxiliary Operations with IBM 1401*

An IBM 1401 Data Processing System can be used for all auxiliary operations. This system consists of a 1401 Processing Unit with as many as 16,000 positions of core storage, a 1402 Card Read Punch, 1403 Printer, and six Magnetic Tape Units (either 729 II or 729 IV). The 1401 is particularly adapted to such conversion operations as: (1) card-to-tape, with off-line audit, control, and edit of input data at 800 cards per minute; (2) tape-to-card, at 250 cards per minute; (3) tape-to-printer, at 600 lines per minute.

Data Synchronizer

In most computers, internal processing speeds are much faster than the input devices that read data or the output devices that record the results. For example, the internal speed of the 709 is approximately 34 times faster than that of the 729 I tape unit and 833 times faster than that of the card reader. However, a word is not always available from tape every 34 machine cycles. Because magnetic tape is a plastic medium being read by a mechanical device, the data transmission rate varies. Actually, input-output devices are asynchronous with the computer; that is, no fixed time relation exists between these devices and the computer.

Some systems overcome this problem by suspending all computing while an I-O device is being used. The core storage waits until the input device has accumulated a data word; this word is stored; the computer then waits for the next piece of data. The stored

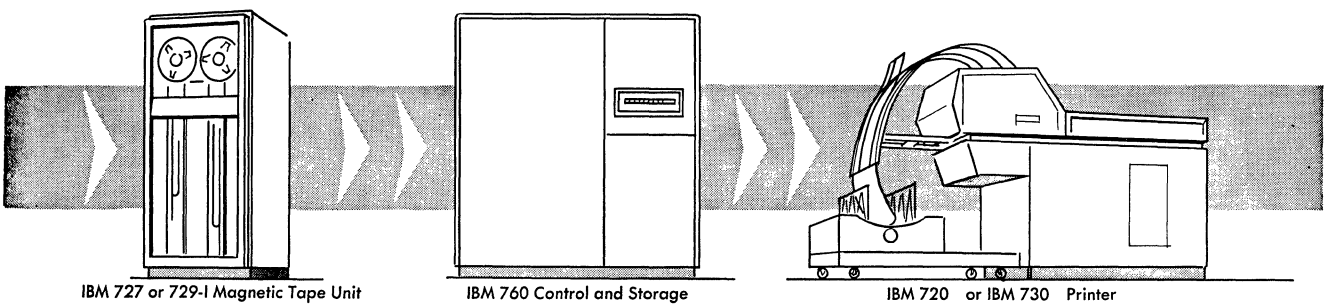


Figure 66. Tape-to-720 or 730 Printer

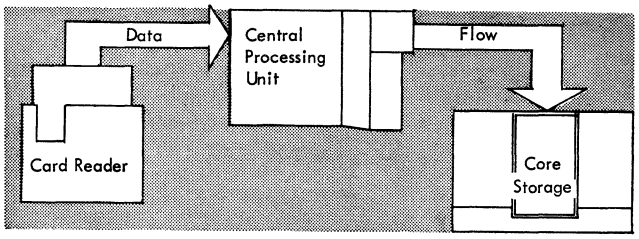


Figure 67. Data Transmission, IBM 704 System

program starts an input operation by defining the input device and the receiving storage area. Processing is delayed while any data are being moved in or out of storage. One advantage to this method is the ease in programming effort. The great disadvantage is the tremendous waste of possible computing time (Figure 67).

The data synchronizers permit the 709 central processing unit to perform a more independent role in controlling the input and output than that of the 704 central processing unit. In the 709 system, the central processing unit initiates and monitors I-O operations but is not interrupted with the detail of routing the data (Figure 68).

Each data synchronizer is composed of two completely separate and independent input-output channels. These two data channels provide for simultaneous computing and transmission between the input-output units and core storage. A data channel may be thought of as a small computer having the responsibility for controlling the quantity and destination of all data transmitted between core storage and an input-output unit. It also performs limited counting and testing operations exclusively concerned with the transmission of data.

No restrictions exist on the type of transmission being performed by a data channel. All channels may be used for input, output, or for a combination of input-output operations that will be concurrent with calculation (Figure 69).

The IBM 766 Data Synchronizer, used with the 709 system, provides a powerful link between core storage and the input-output devices. This new concept of input-output control enables processing to be performed simultaneously with reading and writing. One,

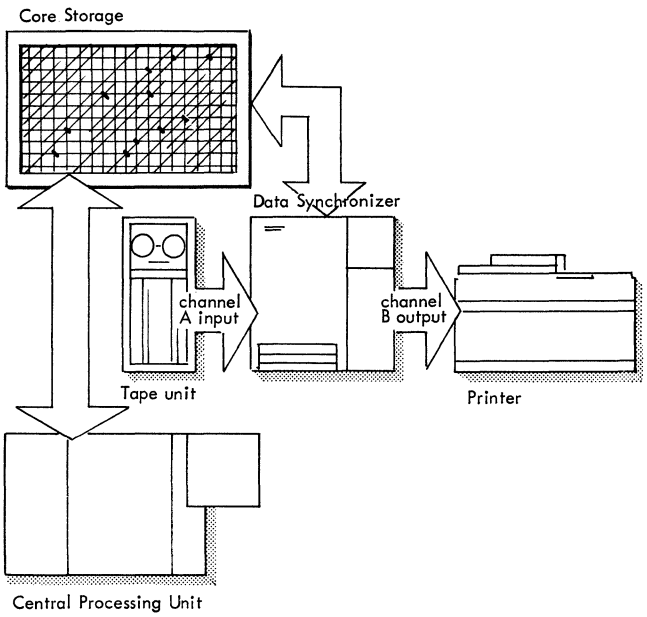
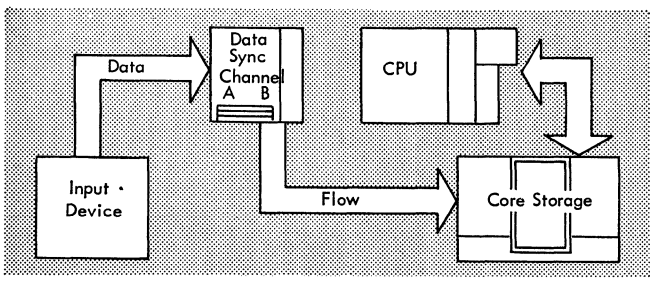


Figure 69. Simultaneous Input, Calculation, and Output

two, or three data synchronizers can be used in the system. Since each data synchronizer contains two input-output data channels, one channel can be reading from a tape while the other channel is writing on another tape, printer, or card punch. At the same time, processing may be executed in the central processing unit.

The IBM 766 Data Synchronizer (Figure 70) provides for improved methods of input-output control listed below. They can be summarized as the ability to:

1. Use the record structure of other IBM input-output devices. This provides for the use of other IBM computer tapes with the 709 system thus making it compatible with other systems.
2. Use any previous IBM computer coding as input data.
3. Transfer input data to scattered core storage locations without loss of programming time.
4. Ignore unwanted data in a record. If desired, unwanted data may be skipped.



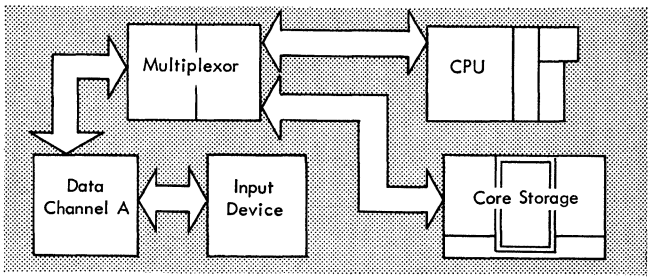


Figure 68. Data Transmission, IBM 709, 7090 Systems

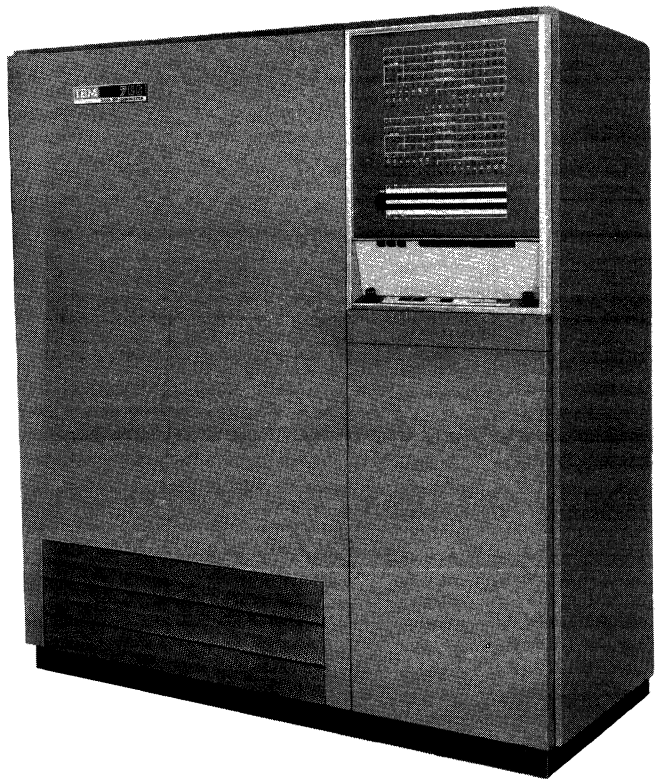


Figure 70. IBM 766 Data Synchronizer

5. Use minimum time in transferring data between input-output units and storage. One core storage cycle is used for each word processed.
6. Synchronize the stored program with the input-output operations when desired. Thus synchronization is optional to the programmer.

To these six refinements, add a seventh and totally new ability to:

7. Operate six different input-output devices and compute simultaneously.

The 709 system programs involving input-output and processing are considered as two separate programs although all instructions are stored in core storage. These two programs are called the processing program and the data channel program.

The processing program is responsible for all arithmetic, logical, and some testing operations. It controls all operations that occur within the central processing unit, plus the starting of input-output operations.

The data channel program is responsible for the actual movement of data, counting of words, and some automatic testing.

The two programs run concurrently and the central processing unit's program is interrupted only when a data channel requires data from, or has data for, core storage.

The operation of a data channel is started by central processing unit instructions. Once started, however, the data channel operates independently of the main program. The data channel is said to contain or "stack" its own instructions. For example:

INSTRUCTION

1. Rewind tape unit 1.
2. Backspace tape unit 2. (These instructions are executed immediately in the central processing unit.)
3. Write tape unit 3.

1. The rewind instruction starts tape unit 1 rewinding. (This may take as long as 1-½ minutes.)

2. Two machine cycles after the rewind instruction is decoded in the central processing unit, the next instruction is decoded. Tape unit 2 then begins its backspace operation. (The time to complete the backspace is a variable depending on the record size.)

3. Again, after two machine cycles, tape unit 3 starts the writing operation. To summarize, the three instructions are executed by the central processing unit in six machine cycles. The central processing unit is then free to process other data until it is asked for a storage reference cycle by the data channel.

Instructions concerned with a data channel are called "commands" instead of instructions in order to keep their understanding apart from the central processing unit instructions.

Figure 71 shows a simple program in which the central processing unit is processing data while the data channel is reading information from a tape. The data channel program, made up of commands, and the instructions for the central processing unit are both stored in core storage. A single command may transmit a large block of words between core storage and an I-O unit. Normally, many instructions in the main program may be executed during the time taken to execute a single command in the data channel.

Main Program. The first main program instruction selects the proper tape attached to data channel A. The second instruction would then load the first command (IOCP) into that data channel. The remaining instructions would then start processing some other central processing unit program.

Main Program Instructions
RTDA ------ read tape decimal, channel A. RCHA------ Reset and load channel A command. central processing unit instructions " " " " " " " "
Data Channel Program Instructions
IOCP ------ I-O count and proceed. IOCPN----- I-O count and proceed, but do not transmit. IORP------- I-O record control and proceed. IOCD------ I-O count and disconnect.

Figure 71. CPU and Data Channel Programs

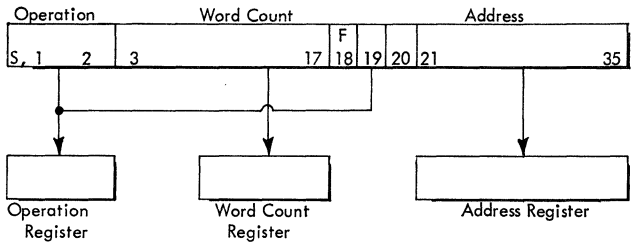


Figure 72. Data Channel Registers

Figure 72 shows the format of data channel commands. The operation part of the command is contained in positions S, 1, 2, and 19.

The address part, controlling the flow of data, is contained in positions 21 through 35.

The count part controls the number of words involved in the operation and is located in positions 3 through 17. Position 18 designates indirect addressing of the command, in the 7090 system only. Position 20 is not used. (Indirect addressing of commands is available for the 709 as an optional feature.)

As can be seen in Figure 72, the count and address parts are put into a word count register and an address register. The operation parts are put into an operation register. These registers are, of course, in the data channel.

Assume that the first command (Figure 71) has an address part of 3000 and a count part of 00006.

INPUT-OUTPUT, COUNT AND PROCEED (IOCP 00006 3000). This command reads the first six words from the selected tape and places them into consecutive core locations starting with location 3000. Each time a word is moved, the count is reduced by one. When the count (00006) is reduced to zero, this command is completed. Because it is a proceed (P) type command, the next command is brought into the data channel for execution.

The second command is INPUT-OUTPUT, COUNT AND PROCEED IN NON-TRANSMIT MODE (IOCPN 00005 0000). This command results in skipping the next five words on the same tape. N designates that the five words involved will be read but not placed in core storage (thus effectively skipping them). Again, because of the P, the next command is brought into the data channel.

The third command, INPUT-OUTPUT UNDER RECORD CONTROL AND PROCEED (IORP 77777 3006), results in reading the remainder of the words in that record into core storage starting with location 3006. Because the number of words remaining in the record is not known, the large count (77777) is used. The sensing of the record gap on the tape signals the end of this command and the next command is brought into the data channel (P).

The fourth command, INPUT-OUTPUT, COUNT AND DISCONNECT (IOCD 00000 0000), disconnects or disengages the data channel, thus signaling the central processing unit that the data channel program is finished.

A maximum of three data synchronizers (six data channels) may be used with a 709 system. The data channels of the first data synchronizer are called A and B, the second two are C and D, and the third two are E and F.

Each data channel may have as many as eight IBM 729 I Magnetic Tape Units attached to it. Data channels A, C, and E may also have one IBM 711 Card Reader, one IBM 721 Card Punch, and one IBM 716 Printer attached to each channel. While all data channels may be operated concurrently with computing, only one input-output unit per channel may be in operation at any given time.

A complete 709 Data Processing System is shown in Figure 73. The additional data synchronizers are not shown but would be attached in the same manner as the first.

To summarize the data synchronizer, the word "channel" is associated with television. Here, a television channel might be defined as a band of frequencies over which one station can broadcast sound, picture, and pulses to synchronize a receiver with the camera. If three television stations are broadcasting simultaneously they must use three separate channels.

A data channel is a group of components that can transmit data between one input or output device and core storage and can synchronize the two. If three I-O devices are to operate simultaneously, they must use three separate channels.

A television receiver can receive only one channel at a time; core storage in a computer can service only one channel at a time. A television transmitter can

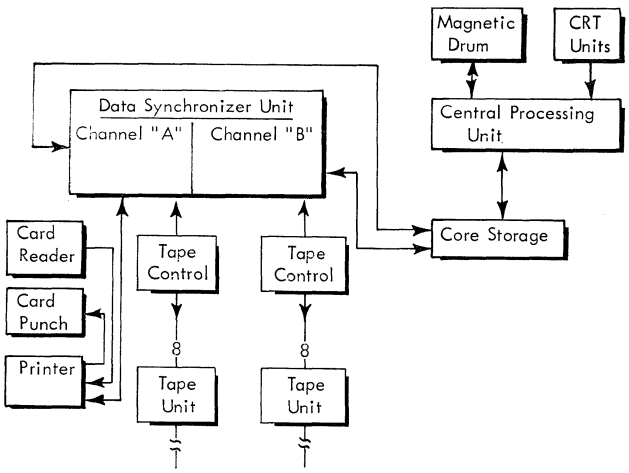


Figure 73. IBM 709 System

broadcast the picture from only one camera at a time; a data channel must use only one I-O device at a time although it is able to control several of them.

With as many as 48 tape units and three sets of card machines controlled by the stored program, the 709 system can perform many jobs "on line" that were formerly performed by auxiliary equipment. On line means that the job is performed using the entire computer system instead of using the "off line" auxiliary equipment.

IBM 755 Tape Control

An IBM 755 Tape Control (Figure 74) is attached between every eight tape units and the appropriate data channel. Several reasons why this is done are:

1. Tape signals must be amplified and analyzed to be usable. Strong noise signals must be rejected, and weak correct signals must be accepted.
2. Only six information bits can be written on tape at one time. The tape control separates the 36-bit word into six 6-bit characters and also generates a seventh checking bit for each character. The characters are then delivered to the tape recording head, one at a time. The process is essentially reversed during tape reading. The data transfer is shown in Figure 74.
3. A signal for the end of record, end of file, weak or invalid character, or the physical end of tape is also passed through the tape control to the data channel.

Only the 704 and 709 systems use tape controls. The IBM 753 Tape Control is used with the 704 and the IBM 755 Tape Control is used with the 709 system.

The problem of asynchronous balance is resolved in the 7090 system through a combination of two new units that perform the simultaneous reading, writing, and computing functions. These units are the IBM 7606 Multiplexor and the IBM 7607 Data Channel.

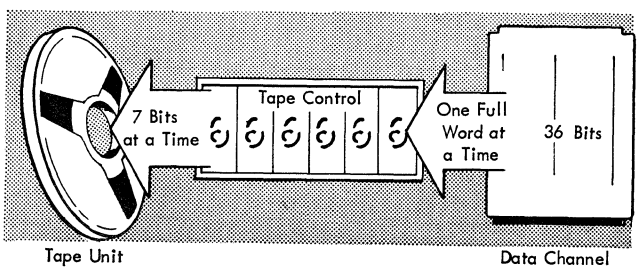


Figure 74. Data Transfer between Tape Control, Tape Unit, and Data Channel

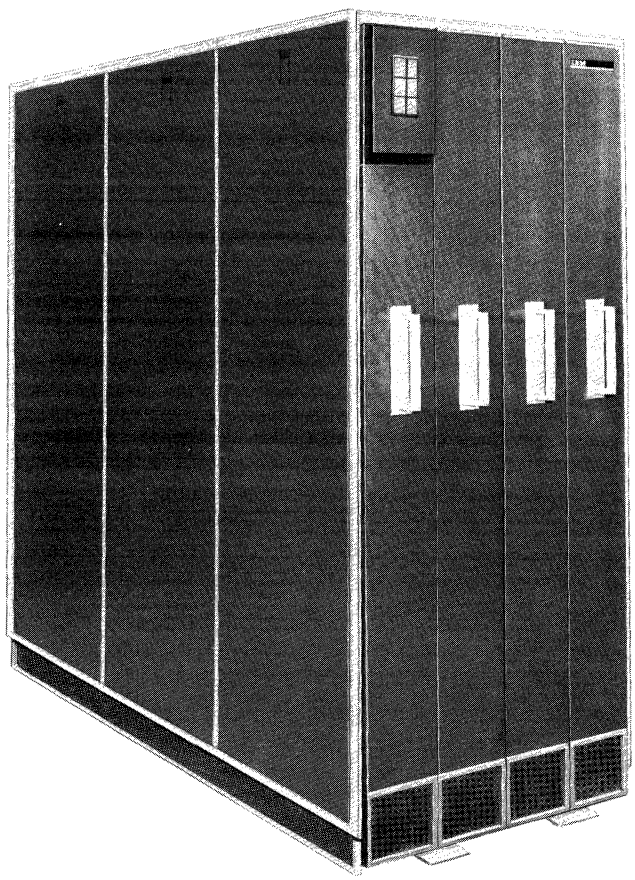


Figure 75. IBM 7606 Multiplexor

Multiplexor

The IBM 7606 Multiplexor (Figure 75) accomplishes all of the data switching in the 7090 system. All input-output components in the system must feed their data through the multiplexor. Likewise, any data coming from core storage must go first to the multiplexor and then to the component.

Data flow in the 7090 system is shown in Figure 76. The data flow path is from an input-output device to the data channel to the multiplexor and then to core storage.

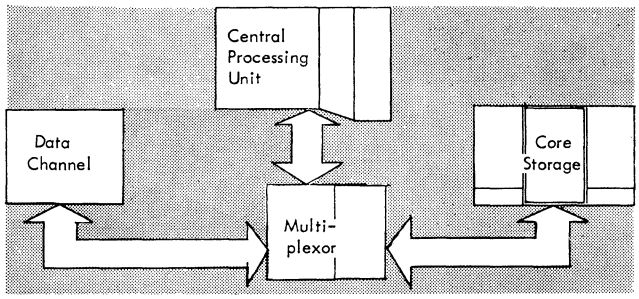


Figure 76. Data Transmission, IBM 7090 System

The multiplexor handles as many as eight input-output data channels. The data word is handled in the normal fashion of 36 bits in parallel, coming from or going to core storage.

Data Channel

The IBM 7607 Data Channel (Figure 77) is another new unit used with the 7090 system. Data channels replace the data synchronizers and tape controls used with the 709 system.

Each data channel is basically a completely separate and independent input-output channel. It provides for the transmission of data between the input-output device and core storage, a transmission that is independent of computing. As in the 709 system, the stored program starts an input operation by defining which input device is to transmit and which core storage area is to receive. The stored program can then proceed to execute instructions (compute) not related to the input transmission. When a full word has been assembled in the data channel, this word is put into core storage. An interruption of the compute program could occur only during the execution of a convert or, in rare cases, some floating point instructions. This interruption could be only for 2.18 microseconds. This procedure is reversed if an output operation is being performed.

Thus, one or more data channels can be occupied with an input operation while others are being used for output purposes. All operations are simultaneous with the processing that is occurring in the central processing unit.

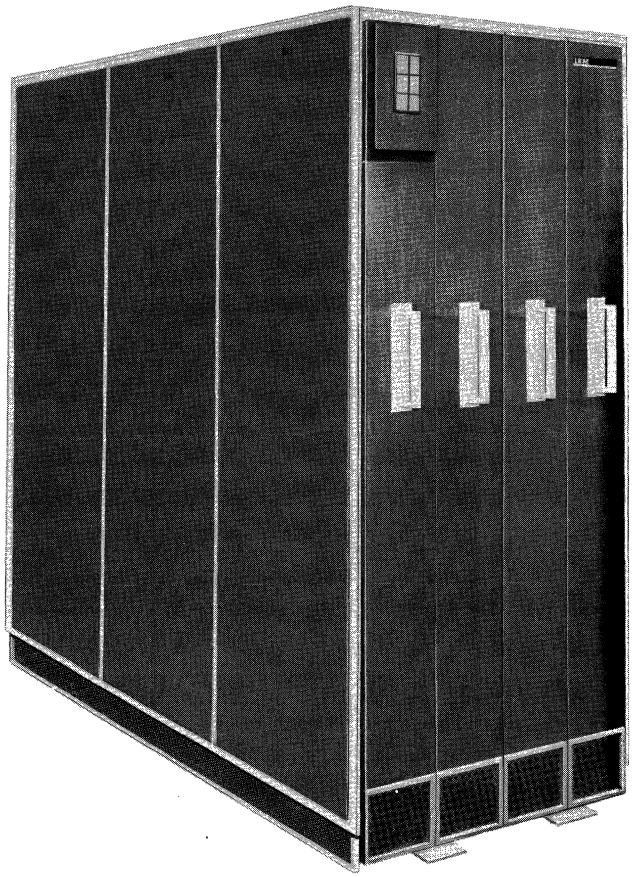


Figure 77. IBM 7607 Data Channel

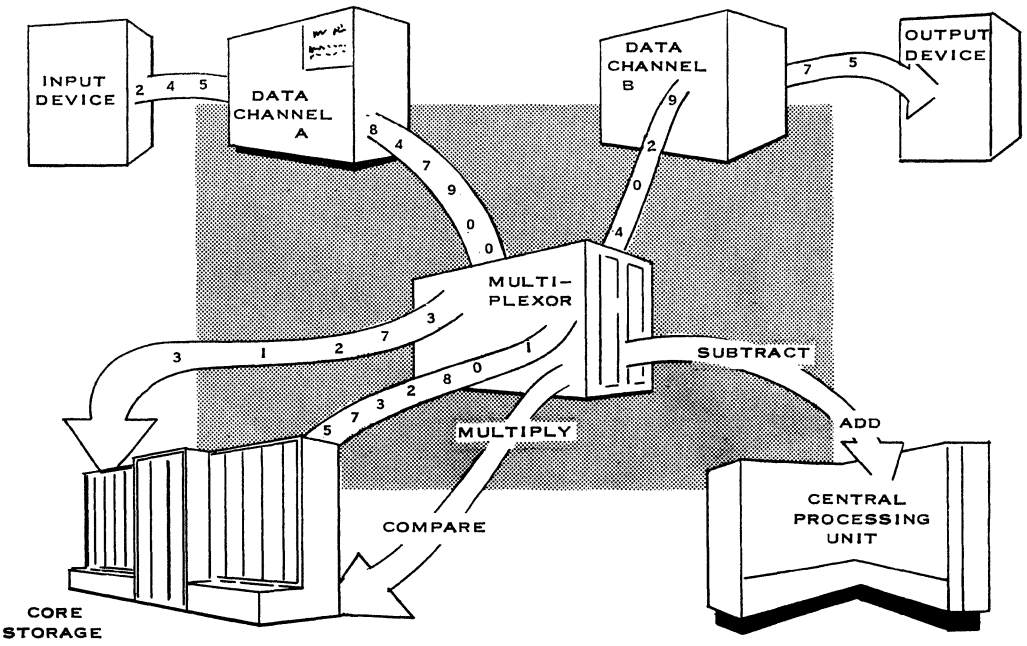


Figure 78. Data Channel and CPU Processing, IBM 7090

A schematic of two data channels operating asynchronously with processing in the central processing unit is shown in Figure 78.

Each data channel may have as many as ten intermixed IBM 729 II and 729 IV Magnetic Tape Units in addition to a card reader, card punch, and a printer.

External Signal

A standard feature of the computer system is its ability to accept a signal from an external source. This signal causes the computer to execute a trapping operation when it is received. The instruction being executed is completed and the location of the next instruction in sequence is placed in the address part of core location 0003. The computer then takes its next instruction from location 0004.

Direct Data Feature

As an optional feature, the computer may be equipped with the direct data feature. This allows the transmission of data between the computer and an external data device. With this feature, an external signal initiates the trapping operation and the stored program may then take whatever action is required to introduce data into the system or supply data from the system to the direct data device.

Thus the main office of a company using the computer could receive data from branch offices by direct wire. This direct data path, as is shown in Figure 79, could be an IBM 65-66 Data Transceiver or any one of many types of transmitting devices.

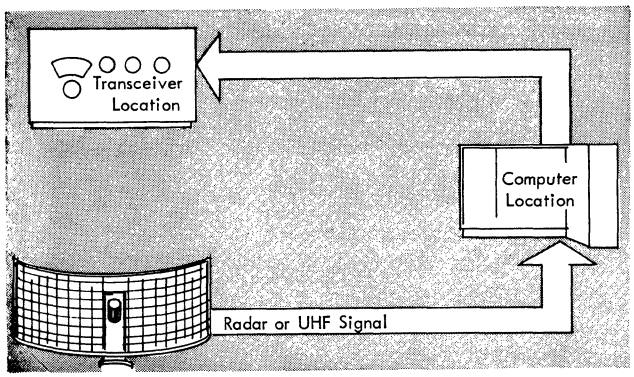


Figure 79. Schematic, External Signal and Direct Data Features

Magnetic Drum Storage

The magnetic drum is another storage device used in electronic computers. It uses the same principle as magnetic tape: A magnetic material can be rapidly magnetized and de-magnetized, and will, like magnetic tape, remain magnetized until deliberately erased. Magnetic drum storage has the advantage that, once information is recorded, it is retained even if the power to the computer is interrupted.

An important additional advantage possessed by a magnetic drum is the ability to read or alter stored information selectively, without reading or rewriting the entire contents of the drum. In the case of magnetic tape, the entire file must be searched or rewritten. This ability to read or write selectively is a valuable asset when large tables are stored and search operations are to be performed or when information stored in the table must be modified. Therefore, the drum is used primarily for the storage of large blocks of related information such as subprograms, rate tables, or supplementary data needed for the solution of a problem.

A comparison between magnetic tape and drum storage shows that tape is a long, thin, storage device using one recording head while the drum is a short, wide, storage device using many recording heads. This similarity is shown in Figure 80.

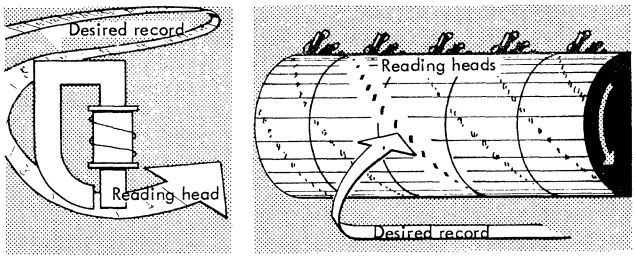


Figure 80. Magnetic Tape and Drum as Storage Devices

The magnetic material used on the drum is a wire made of an alloy containing copper, nickel, cobalt, and iron. This wire is wound and fused to the surface of the drum and the surface is then ground to an extremely fine tolerance, providing a smooth even magnetic surface.

The writing or recording process is similar to magnetic tape in that a coated surface passes under a recording head. Information is written on this magnetic surface by an electrical pulse that causes the head to become active. Whenever the head is active, the area directly under it becomes magnetized as shown in Figure 81. This area is called a bit or a one. Thus, information is stored on the surface of the drum in a pattern represented by combinations of magnetized areas.

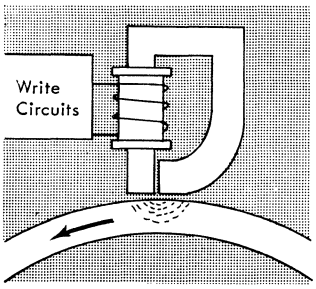


Figure 81. Writing on a Drum

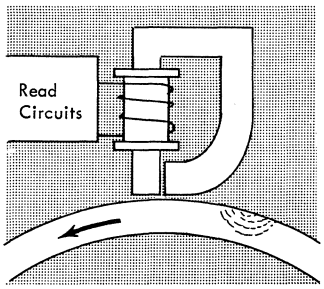


Figure 82. Reading from a Drum

To read the information, the magnetized surface is passed under the recording head. This time, the magnetized area on the drum surface generates small voltage pulses in the head as shown in Figure 82, exactly the reverse of the writing process. These voltage pulses are amplified and are then available for use in the computer.

The 36 recording heads spread across the surface of one drum accommodate the word size of the computer (Figure 83).

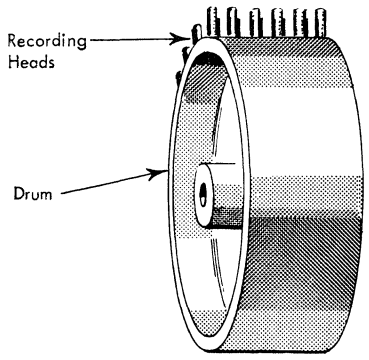


Figure 83. Magnetic Drum Recording Heads

A magnetic drum unit has a storage capacity of 8,192 words, each word consisting of 36 bits. The drum unit contains two distinct physical drums each with a capacity of 4,096 words. Each of the physical drums contains two logical drums with a capacity of 2,048 words each. A logical drum is selected by giving the appropriate address.

The 2,048 locations on each logical drum can be individually addressed by integers in the range 0000-2047 (0000-3777 octal). A record or block of words

Figure 85. IBM 733 Magnetic Drum

is normally stored in sequentially numbered locations. The program must indicate the drum address where the first word is to be written or read. The number of copy instructions executed then determines the number of words in the record.

Figure 84 illustrates the physical arrangement of words on a logical drum. The addresses are numbered octally. Observe that, when reading or writing a continuous record, the computer refers to every eighth word of a drum for consecutive addresses. Each logical drum has 256 sectors (a group of eight locations) and therefore must make eight complete revolutions for all 2,048 words to be read or written as a continuous record.

In summary, the drum is a high-volume, moderate-speed, long-term data storage medium for use with both the 704 and the 709 systems. It is particularly useful for the storage of large blocks of information such as subroutines, rate tables, or supplementary data needed for the solution of a problem. The data stored on the drum are not destroyed when power is re-

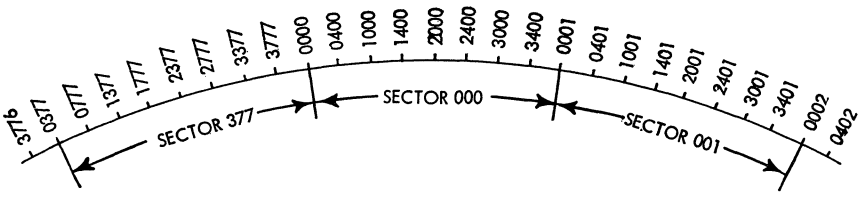


Figure 84. Drum Sectors and Addresses

moved. This makes the drum valuable for overnight storage of intermediate results during the processing of involved problems.

The IBM 733 Magnetic Drum (Figure 85) is one of two input-output units in the 709 system that function without being dependent upon the data synchronizer. The other unit is the IBM 740 CRT Output Recorder and Display Unit. The 7090 system does not use the magnetic drum, or any of the CRT equipment.

Punched Cards

In most applications, magnetic tape is used as the principal input medium. It may also be desirable to use IBM cards as an input medium in some situations where the volume of input is sufficiently small to permit an economical operation. In either case, IBM cards are used for initially recording data because of their great flexibility. Errors are easily detected and corrected, input data may be readily prepared on several card punch locations simultaneously and the cards may be collected at a central location before entry into the computer.

Entering a program on cards may be done in such a way that instructions are punched, one to a card, in the most desirable form to the programmer. The computer can then be supplied with a standard program to assemble the instructions in the desired order. Then, if errors are detected or if changes must be made, the error cards are removed, the correct ones are added, and the computer prepares the new program. Note that there is no need to repunch all cards but only the cards in error.

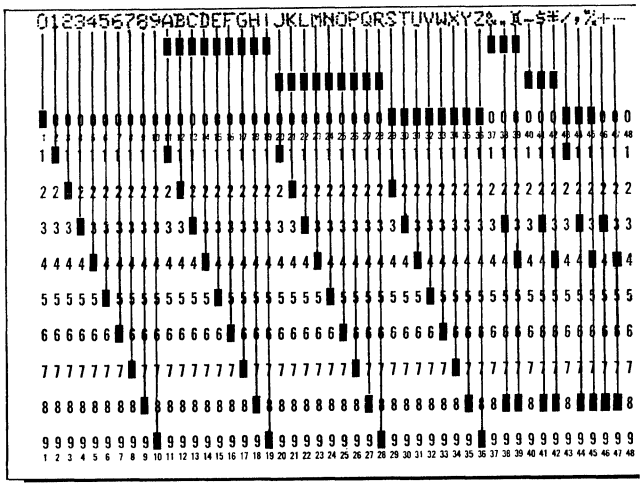


Figure 86. Standard IBM Card Code

Punched card input and output may represent any alphabetic character, special symbol, or binary punching if the assembly program handling it is able to recognize the code used. The standard IBM card code is shown in Figure 86.

Only 72 card columns of the IBM card can be read into or punched from core storage during any given storage reference cycle. Any 72 of the possible 80 card columns of the card can be selected through use of a control panel on the card reader and the proper wiring on that panel. Figure 87 shows how the card may be punched to record data in binary form. In this particular example, the first 72 columns of the card are used. Each row is split into half-rows of 36 card columns, each corresponding to the 36-bit binary word. Thus, the half-row identified by the circled 9 is named the 5-row left. Similarly, the row identified by the circled 10 is named the 5-row right.

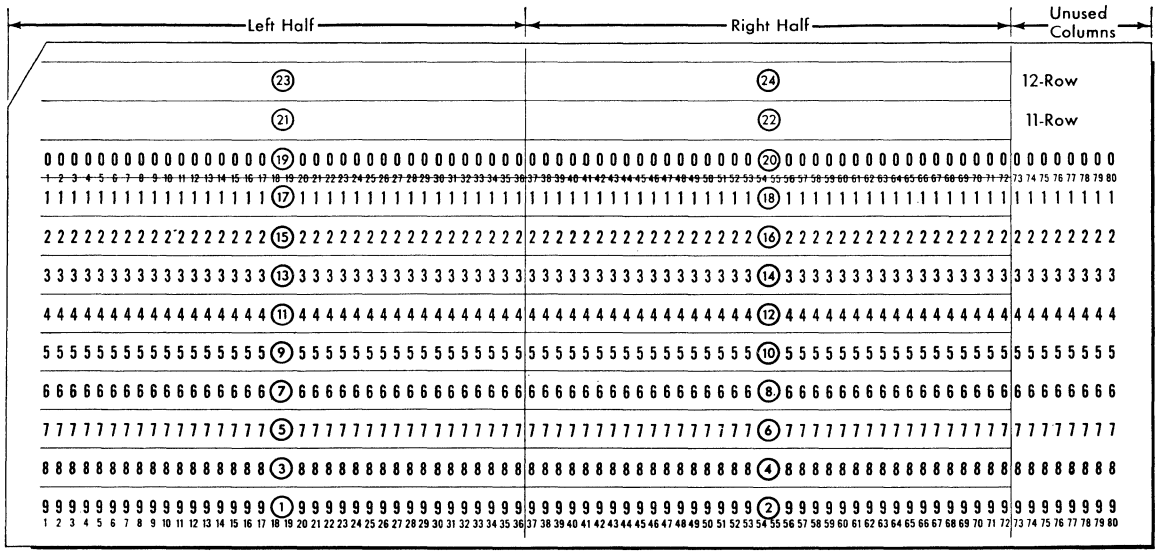


Figure 87. Sequence of Card Reading

Thus, there are 24 half-rows in the card. One full word of binary data (including the sign) can be punched in any half-row. The card reader regards the punched hole as a binary 1. An unpunched hole indicates a binary 0. For example, a punched hole in the 8-row, card column 36, is regarded by the reader as a binary 1 in the least significant position of the word in the 8-row left. The left-most position of each half row is reserved for the sign bit of that word (columns 1 and 37). A binary 1 represents a negative sign while a binary 0 (no punch) represents a positive sign.

When reading or punching cards, a *record* is defined as the information contained in one card. A *file* consists of any number of cards and takes the form of a *deck*. Note that the definitions of records and files differ depending on the particular input-output component being discussed.

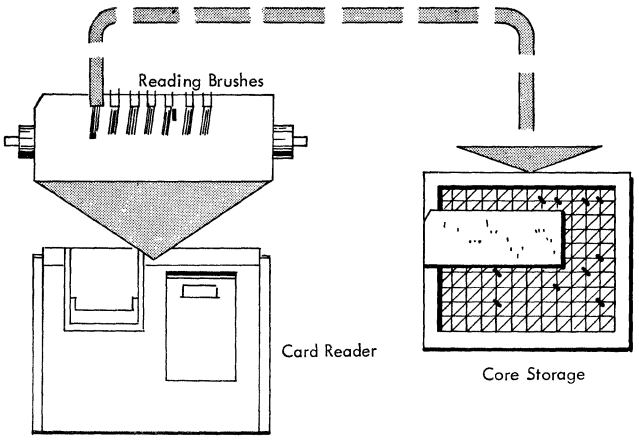


Figure 89. Card Image in Core Storage

Card Reader

The IBM 711 Card Reader (Figure 88) is used by the computer as the card input device. This reader is available in two models. The model I reads cards at the rate of 150 per minute while the model II reads 250 cards per minute.

Data punched in the cards may be numerical, alphabetic, binary, or any special character code. The interpretation of the coding and reading format is controlled by the stored program and the control panel located on the reader. As a punched card proceeds through the reader, a "card image" is built up in core storage (Figure 89). As additional cards are read, they too become a part of the card images stored in core storage.

The IBM 709 Data Processing System may have one model II card reader attached to data synchronizer channel A, and additional readers may be attached to channels C and E. If a reader is attached to a data channel, an IBM 716 Printer must also be attached to the same channel because it is the power source for the reader.

The IBM 7090 Data Processing System may have one model II reader attached to each data channel, making a total of eight possible readers on the system. A printer is required for each reader.

Figure 88. IBM 711 Card Reader

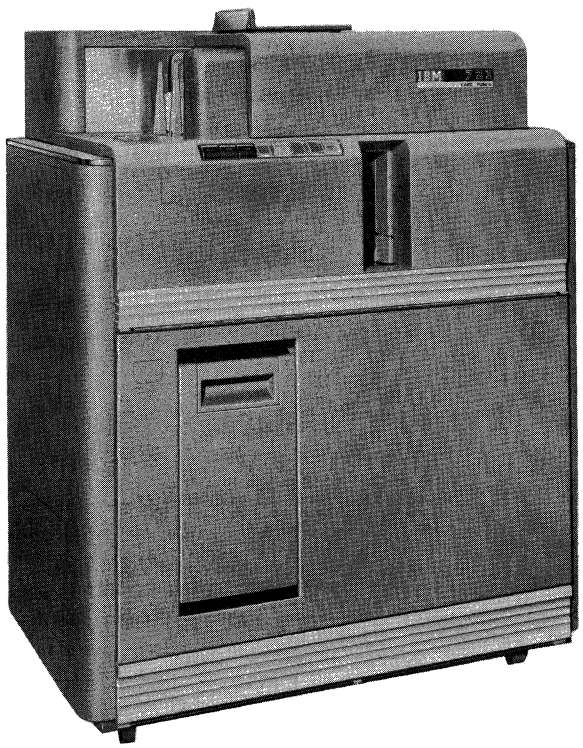


Figure 90. IBM 721 Card Punch

Card Punch

The IBM 721 Card Punch (Figure 90) is used as the punching or recording unit for IBM cards. The 721 punches cards at a rate of 100 cards per minute.

Basic card punch operations are analogous to those of the card reader, except that instead of building up a card image in core storage when data are read from a card, the card image is sent from core storage to the card punch to be recorded as punched holes (Figure 91). The stored program and the control panel on the card punch control the coding and the recording format of the card image.

The IBM 709 Data Processing System may have one card punch attached to data channels A, C, and E. If a card punch is attached to a data channel, that channel must also have an IBM 716 Printer attached to it. One printer will accommodate both the reader and punch.

The IBM 7090 Data Processing System may have one card punch attached to each data channel making a total of eight possible card punches per system. One printer is required for each card reader and card punch.

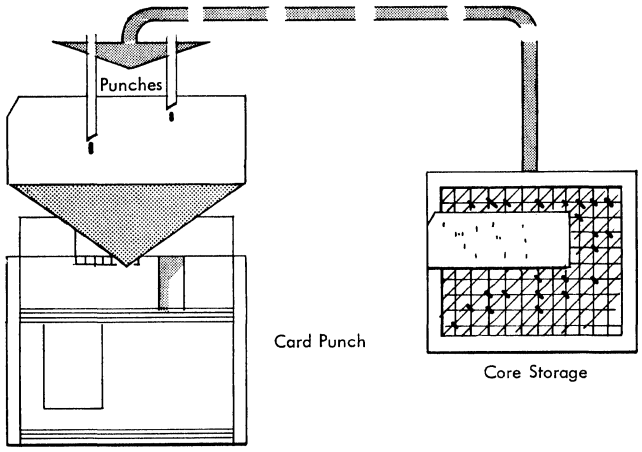


Figure 91. Punching a Card Image

Printer

The IBM 716 Printer (Figure 92) is used to prepare printed output from the computer systems.

The printer is equipped with 120 rotary type wheels (Figure 93). Each type wheel has 48 characters in-

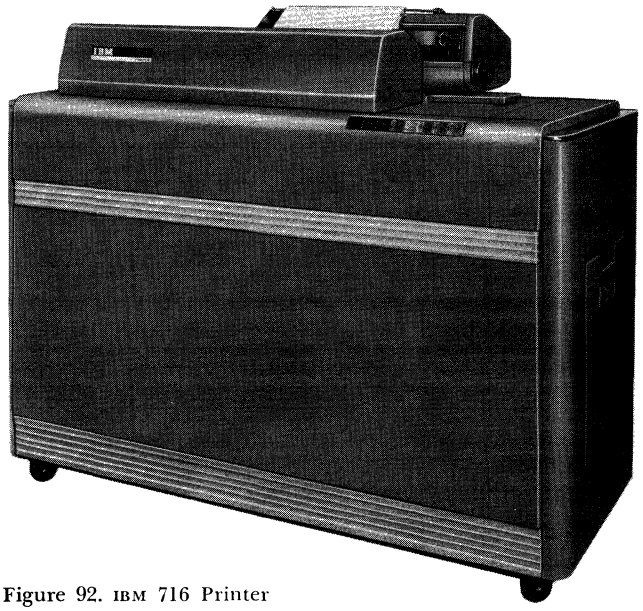


Figure 92. IBM 716 Printer

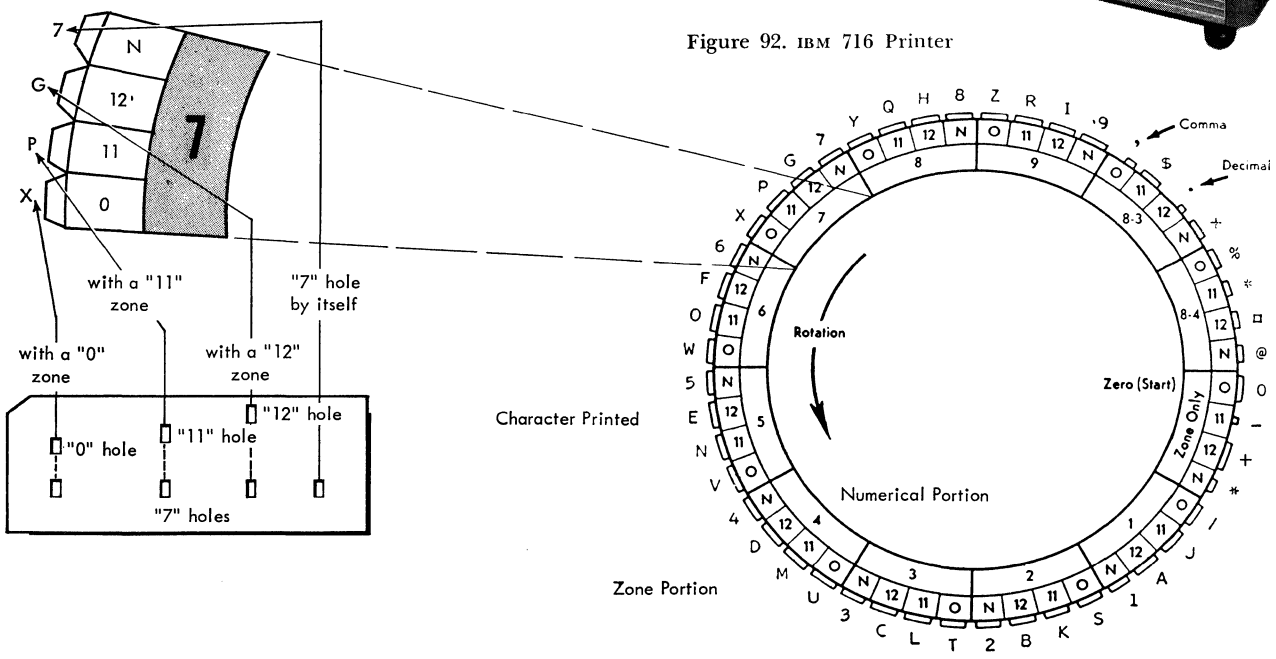


Figure 93. IBM 716 Printer Type Wheel

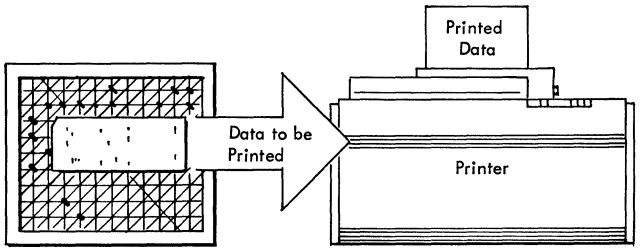


Figure 94. Printing from Core Storage

cluding numerals, alphabetic symbols, and special characters. Information is printed at the rate of 150 lines per minute with as many as 120 characters per line. The print format and arrangement of the data to be printed are controlled by the stored program and a control panel located on the printer. Use of the proper stored program enables the printing of any desired information in any form convenient to the programmer.

Figure 94 shows a report as it appears in core storage and as it would appear being printed on the 716 Printer. The choice of which type wheels to use in the actual printing would be decided by the control panel wiring on the printer.

The IBM 709 Data Processing System uses one printer attached to data channels A, C, and E. The IBM 7090 Data Processing System may have a total of eight printers, one attached to each data channel. With all systems, the 716 Printer supplies the power for both the card reader and the card punch and, therefore, each set of card equipment must be attached to one data channel.

Cathode Ray Tube Equipment

Visual display of processed data is provided by a cathode ray tube display unit. Numbers are converted into alphamerical characters, curves, or any type of visual representation. With this unit, the actual results determined by the mathematical formula used in Figure 1 could be used to display the cross section of the wing design (Figure 95).

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

When a character is to be displayed, only 35 positions of the word containing the character are used. The sign position is considered positive. The remaining bits signify a display or no-display of a spot, de-

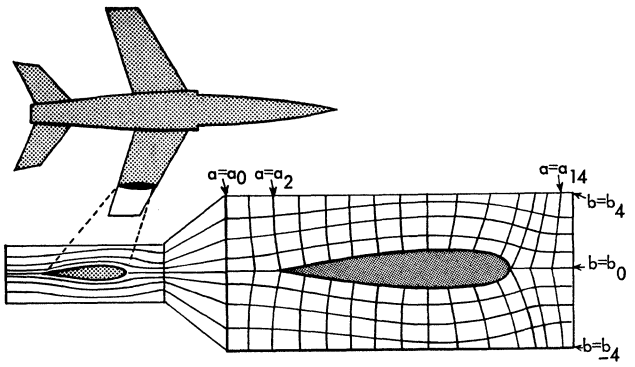


Figure 95. Visual Display of Wing Design

pending on whether the bit is a 1 or a 0. The first column of the character is represented by the first group of seven bits (following the sign bit); the second column, by the second group of 7 bits, and so on. The first bit of each group may correspond to the lowest position of each column; succeeding bits will move up the column, one bit at a time.

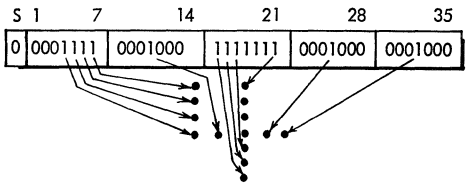


Figure 96. CRT Character Plotting

An example of CRT character plotting is shown in Figure 96. The word responsible for the plot, as it appears in core storage, is shown above the plot. The program consists of a write instruction addressing the CRT and a copy instruction specifying the storage location of the word containing the character to be plotted.

The IBM 740 Cathode Ray Tube Recorder (Figure 97) is basically a digital-to-analog converter. It is connected to both the 704 and 709 systems. The recorder contains a seven-inch cathode ray tube as the actual output device. A camera, mounted over the face of the CRT, produces a filmed recording of the plotted data.

The IBM 780 Cathode Ray Tube Display (Figure 98) is connected to the CRT recorder and controlled by it. This unit contains a 21-inch CRT, similar to a home television receiver tube, to provide an immediate display of the plot being filmed by the 740 unit. The CRT display is, in effect, a "slave unit" to the CRT recorder. The conversion from digital to analog data,

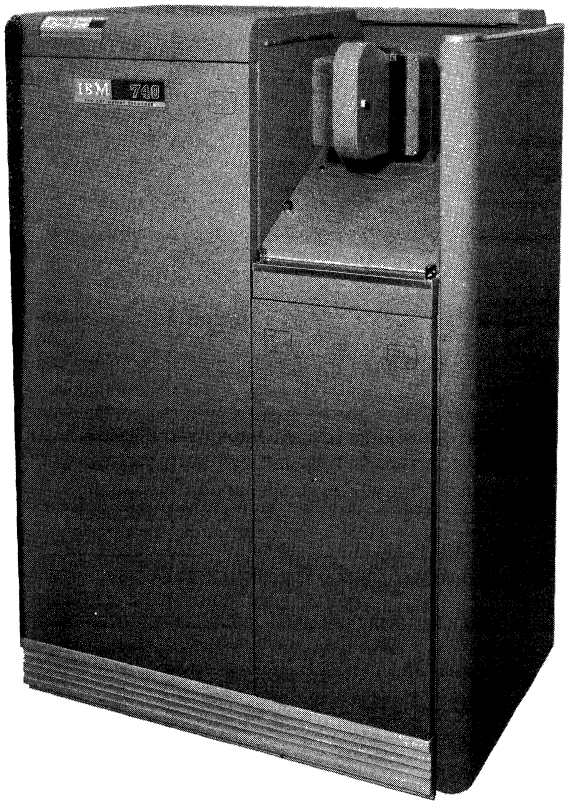


Figure 97. IBM 740 CRT Output Recorder

the display of that data on the faces of the CRT's, and the operation of the camera are all accomplished at electronic speeds under control of the stored program.

The 7090 system does not use either the CRT recorder or the CRT display units.

Figure 99 shows comparative statistics of the 709 and 7090 systems. Refer to the *Reference Manuals* (*IBM 709*, Form A22-6536, or *IBM 7090*, Form A22-6528) for the operational details concerning the systems.

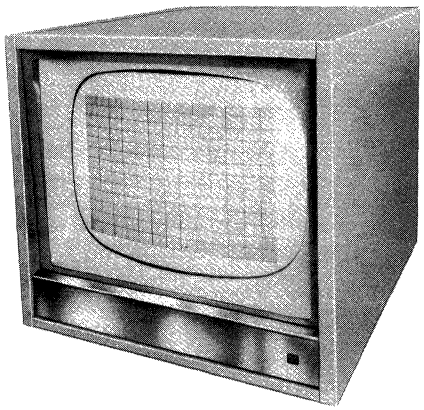


Figure 98. IBM 780 CRT Display

	709	7090
Core Storage Size	4096, 8192 or 32,768 words	32,768 words
Transistorized	no	yes
Internal Speed (Basic Cycle)	12 usec	2.18 usec
Simultaneous Read-Write-Compute	yes	yes
Tape Skip Ability	yes	yes
Automatic Input-Output Priority	yes	yes
Data Channel Trapping	yes	yes
Number of Instructions	208	220
Maximum Number of:		
Tape Units	48	80
I-O Data Channels	2,4 or 6	1 thru 8
Card Readers	3	8
Card Punches	3	8
Printers	3	8
Magnetic Drums	2	none
CRT Recorder and Display	1	none

Figure 99. Comparison of the IBM 709 and 7090 Systems

Share

Organization

The present Share organization was conceived when three customers, involved in preparation for a 704 Data Processing System, began informal discussions concerning their individual plans and problems. The mutual respect of the participants for the programming competence of each other was expressed; they were willing to accept the ideas of others, even to the extent of obsoleting methods already prepared within their own installations. It was agreed that a full-scale attempt should be made to bring Share into being. The first meeting was held in 1955 with seventeen members present.

One of the advantages of an organization of this type is that each member is closely united with the development of computer usage throughout the world. A substantial portion of the major users of high-speed digital computers is represented in the Share membership. It has been found that critical evaluation of each other's ideas usually produces a distillation of thoughts, superior to any individual opinions.

Members realize substantial savings in programming and check-out of programs. The continual interchange of ideas among members has demonstrated that a high degree of computing sophistication is rapidly built up in a Share installation.

In its initial phase, Share was concerned primarily with procedures and standards. Later, the preparation and distribution of programs was started. No slacking of activity is anticipated since, as the most essential programs are completed, emphasis shifts to new areas of mathematical computation.

Programming System

The Share 709 system is used with the 7090 Data Processing System as the main programming system. It enables the programmers to write, check-out, and alter their programs quickly and easily. The Share 709 System (sos) can be thought of as composed of four distinct parts:

1. The Share compiler assembler translator (SCAT).
2. The program testing and correcting system.
3. The input-output system.
4. The MockDonald control system.

The SCAT portion of the sos consists of two parts, the compiler and the modify-and-load program.

The compiler performs about the same functions for the 709 and 7090 systems that the Share assembly program (SAP) performs for the 704 system. With a few minor exceptions, a Share assembly language program is acceptable as input for the compiler and results in a program listing and an absolute binary program card deck. In addition to the results produced by Share assembly program, the compiler can produce a "squoze" card deck. This deck contains the symbolic source program in encoded binary form. This form may be converted to machine language and loaded by the modify and load program almost as rapidly as an ordinary binary load program loads an absolute binary card deck. Two main reasons for the intermediate squoze card deck phase are:

1. Modifications to the program can be made in the original SCAT language and then added to the squoze deck for loading by the modify-and-load program.
2. Enough of the original symbolic information can be retained during the program execution to permit the checking and correcting program to return printed output in the original symbolic language. In most instances, these two features make it unnecessary to "patch" a program in machine language. Thus, nearly all cross-referencing between symbolic and binary may be avoided.

Another powerful tool in the compiler is the idea of "macro-operation." The compiler is built to recognize a large fixed number of macro-operations. It also accepts and temporarily retains definitions of macro-operations given by the programmer. In either case, it generates and inserts into the program the sequence of machine words specified by any one of these macro-operations in a macro-instruction. Among the system (fixed) macros are all of the pseudo-operations making up the checking and correcting program, the input-output, and the MockDonald control system languages.

A significant feature of the SCAT system is that the loading process is also an assembly process. The squoze deck is not in a form that can be inserted into the computer as is. It is the result of what corresponds to the first pass of the 704 Share assembly program system. With SAP a second pass is needed to decode the assembled deck into absolute binary. In the SCAT system, the second pass is a function of the modify-and-

load program. This program provides for the same modification of the original code that could be obtained by changing the original symbolic deck and reassembling. Thus, in SCAT, program modifications are given to the loading program and, for these changes, the modify-and-load program performs both the first and second passes. In this sense, the loading program is a full assembly program. When program modifications are made with the loader, it produces on request a new squoze deck or an absolute binary deck, as well as the listing of the modified program. In addition to the above operations, the modify-and-load program performs checking operations for both programmer and machine errors.

The input-output system permits writing of I-O programs designed for a particular customer's application. Transmission macro-instructions are used and are executed by a routine called the *dispatcher*. These macros provide simultaneous input-output with computing. Computing is interlocked with data transmission so that the computer will not attempt to use or modify data to be completed. Transmission orders are channel-stacked when required and subsequently the dispatching of these orders on data channels is automatic when the channel is free. The programmer may interrogate the dispatcher for the status of any transmission at any point in the program. The checking of input-output indicators is automatically accomplished by the dispatcher.

The MockDonald control system has been designed to enable the automatic transition from one problem to the next, to maintain a machine program log, to aid in the parallel operation of tapes and main frame processing and, in general, to perform many of the operations that would normally be handled by a professional machine operator.

Fortran Automatic Coding System

The IBM Mathematical FORmula TRANslating System, 709 FORTRAN, is an automatic coding system for the IBM 709 Data Processing System. More precisely, it is a 709 program that accepts a *source program* written in the Fortran language, closely resembling the ordinary language of mathematics, and produces a machine language *object program* ready to be run on a 709.

The 709 Fortran therefore, in effect, transforms the IBM 709 into a machine with which communication can be made in a language more concise and more familiar than the 709 machine language itself. The result is a substantial reduction in the training required to program, as well as in the time consumed in writing programs and eliminating errors from them.

Among the features which characterize the 709 Fortran system are the following:

The 709 Fortran is available in versions for all sizes of storage. Each version produces programs which can be used on any size of 709, provided sufficient storage is available for the object program. Object programs that are too large for the 709 on which they are to be used must be subdivided by the user.

Object programs produced by Fortran will generally be as efficient as those written by experienced programmers.

The Fortran language is intended to provide facilities for expressing any problem of numerical computation. In particular, problems containing large sets of formulas and many variables can be dealt with easily, and any variable may have up to three independent subscripts.

The language of Fortran may be expanded by the use of subprograms. These subprograms may be written in Fortran language, and may be called by other Fortran programs, as well as subprograms. The language may be expanded by the use of subprograms to any desired depth.

For problems in which machine words have a logical rather than a numerical meaning, the language is less satisfactory, and difficulties may arise in expressing such problems. Nevertheless, many logical operations not directly expressible in the Fortran language can be carried out by making use of the provisions for incorporating library routines.

Prewritten routines, to evaluate functions of any number of arguments, can be made available for incorporation into object programs by the use of any of several different facilities provided for this purpose.

Certain statements in the Fortran language cause the inclusion in the object program of the necessary input and output routines. Those which deal with decimal information include conversion to or from the internal machine language, and permit considerable freedom of format in data input and output.

Arithmetic in an object program will generally be performed with single-precision floating point numbers. These numbers provide about 8 decimal digits of precision, and may be zero or have magnitudes between approximately 10^{-38} and 10^{38}. Fixed point arithmetic for integers is also provided.

SCAT Mnemonic Operation Codes

CODE	COMMENT	INDEXABLE	IND. ADDRESS
ACL	Add and Carry Logical Word	X	X
ADD	Add	X	X
ADM	Add Magnitude	X	X
ALS	Accumulator Left Shift	X	
ANA	AND to Accumulator	X	X
ANS	AND to Storage	X	X
ARS	Accumulator Right Shift	X	
AXC	Address to Index, Complemented		
AXT	Address to Index, True		
BSFA	Backspace File, Ch. A	X	
BSFB	Backspace File, Ch. B	X	
BSFC	Backspace File, Ch. C	X	
BSFD	Backspace File, Ch. D	X	
BSFE	Backspace File, Ch. E	X	
BSFF	Backspace File, Ch. F	X	
**BSFG	Backspace File, Ch. G	X	
**BSFH	Backspace File, Ch. H	X	
BSR	Backspace Record	X	
BSRA	Backspace Record, Ch. A	X	
BSRB	Backspace Record, Ch. B	X	
BSRC	Backspace Record, Ch. C	X	
BSRD	Backspace Record, Ch. D	X	
BSRE	Backspace Record, Ch. E	X	
BSRF	Backspace Record, Ch. F	X	
**BSRG	Backspace Record, Ch. G	X	
**BSRH	Backspace Record, Ch. H	X	
BTTA	Beginning of Tape Test, Ch. A	X	
BTTB	Beginning of Tape Test, Ch. B	X	
BTTC	Beginning of Tape Test, Ch. C	X	
BTTD	Beginning of Tape Test, Ch. D	X	
BTTE	Beginning of Tape Test, Ch. E	X	
BTTF	Beginning of Tape Test, Ch. F	X	
**BTTG	Beginning of Tape Test, Ch. G	X	
**BTTH	Beginning of Tape Test, Ch. H	X	
*CAD	Copy and Add Logical Word	X	
CAL	Clear and Add Logical Word	X	X
CAQ	Convert by Addition from MQ		
CAS	Compare Accumulator with Storage	X	X
CFF	Change Film Frame	X	
CHS	Change Sign	X	
CLA	Clear and Add	X	X
CLM	Clear Magnitude	X	
CLS	Clear and Subtract	X	X
COM	Complement Magnitude	X	
*CPY	Copy	X	
CRQ	Convert by Replacement from MQ		
CVR	Convert by Replacement from AC		
DCT	Divide Check Test	X	
DVH	Divide or Halt	X	X
DVP	Divide or Proceed	X	X
ENK	Enter Keys	X	
ERA	Exclusive OR to Accumulator	X	X
ETM	Enter Trapping Mode	X	
ETTA	End of Tape Test, Ch. A	X	
ETTB	End of Tape Test, Ch. B	X	
ETTC	End of Tape Test, Ch. C	X	
ETTD	End of Tape Test, Ch. D	X	
ETTE	End of Tape Test, Ch. E	X	
ETTF	End of Tape Test, Ch. F	X	
**ETTG	End of Tape Test, Ch. G	X	
**ETTH	End of Tape Test, Ch. H	X	
FAD	Floating Add	X	X
FAM	Floating Add Magnitude	X	X
FDH	Floating Divide or Halt	X	X
FDP	Floating Divide or Proceed	X	X
FMP	Floating Multiply	X	X
FOR	Four		
FRN	Floating Round	X	
FSB	Floating Subtract	X	X
FSM	Floating Subtract Magnitude	X	X
FVE	Five		
HPR	Halt and Proceed		
HTR	Halt and Transfer	X	X
IIA	Invert Indicators from Accumulator		
IIL	Invert Indicators of the Left Half		
IIR	Invert Indicators of the Right Half		
IIS	Invert Indicators from Storage	X	X
IOCD	Input-Output under Count Control and Disconnect		
IOCDN	(IOCD with No Transmission)		
IOCP	Input-Output under Count Control and Proceed		
IOCPN	(IOCP with No Transmission)		
IOCT	Input-Output under Count Control and Transfer		
IOCTN	(IOCT with No Transmission)		
IORP	Input-Output of a Record and Proceed		
IORPN	(IORP with No Transmission)		
IORT	Input-Output of a Record and Transfer		
IORTN	(IORT with No Transmission)		
IOSP	Input-Output until Signal and Proceed		
IOSPN	(IOSP with No Transmission)		
IOST	Input-Output until Signal and Transfer		
IOSTN	(IOST with No Transmission)		
IOT	Input-Output Check Test	X	
LAC	Load Complement of Address in Index		
LAS	Logical Compare Accumulator with Storage	X	X
LBT	Low-Order Bit Test	X	
LCHA	Load Channel A	X	X
LCHB	Load Channel B	X	X
LCHC	Load Channel C	X	X
LCHD	Load Channel D	X	X
LCHE	Load Channel E	X	X
LCHF	Load Channel F	X	X
**LCHG	Load Channel G	X	
**LCHH	Load Channel H	X	
*LDA	Locate Drum Address	X	X
LDC	Load Complement of Decrement in XR		
LDI	Load Indicators	X	X
LDQ	Load the MQ	X	X
LFT	Left Half Indicators, Off Test		
LGL	Logical Left Shift	X	
LGR	Logical Right Shift	X	
LLS	Long Left Shift	X	
LNT	Left Half Indicators, On Test		
LRS	Long Right Shift	X	
LTM	Leave Trapping Mode	X	
LXA	Load Index from Address		
LXD	Load Index from Decrement		
MON	Minus One		
MPR	Multiply and Round	X	X
MPY	Multiply	X	X
MSE	Minus Sense	X	

*709 system instruction only.
**7090 system instruction only.

SCAT Mnemonic Operation Codes *(Cont'd)*

CODE	COMMENT	INDEXABLE	IND. ADDRESS
ORA	OR to Accumulator	X	X
ORS	OR to Storage	X	X
OSI	OR Storage to Indicators	X	X
PAC	Place Complement of Address in XR		
PAI	Place Accumulator in Indicators		
PAX	Place Address in Index		
PBT	P-Bit Test	X	
PDC	Place Complement of Decrement in XR		
PDX	Place Decrement in Index		
PIA	Place Indicators in Accumulator		
PON	Plus One		
PSE	Plus Sense	X	
PTH	Plus Three		
PTW	Plus Two		
PXA	Place Index in Address		
PXD	Place Index in Decrement		
PZE	Plus Zero		
RCDA	Read Card Reader, Ch. A	X	
**RCDB	Read Card Reader, Ch. B	X	
RCDC	Read Card Reader, Ch. C	X	
**RCDD	Read Card Reader, Ch. D	X	
RCDE	Read Card Reader, Ch. E	X	
**RCDF	Read Card Reader, Ch. F	X	
**RCDG	Read Card Reader, Ch. G	X	
**RCDH	Read Card Reader, Ch. H	X	
RCHA	Reset and Load, Ch. A	X	X
RCHB	Reset and Load, Ch. B	X	X
RCHC	Reset and Load, Ch. C	X	X
RCHD	Reset and Load, Ch. D	X	X
RCHE	Reset and Load, Ch. E	X	X
RCHF	Reset and Load, Ch. F	X	X
**RCHG	Reset and Load, Ch. G	X	X
**RCHH	Reset and Load, Ch. H	X	X
** RDCA	Reset Data Channel A	X	
RDCB	Reset Data Channel B	X	
RDCC	Reset Data Channel C	X	
RDCD	Reset Data Channel D	X	
RDCE	Reset Data Channel E	X	
RDCF	Reset Data Channel F	X	
RDCG	Reset Data Channel G	X	
** RDCH	Reset Data Channel H	X	
*RDR	Read Drum	X	
RDS	Read Select	X	
REWA	Rewind, Ch. A	X	
REWB	Rewind, Ch. B	X	
REWC	Rewind, Ch. C	X	
REWD	Rewind, Ch. D	X	
REWE	Rewind, Ch. E	X	
REWF	Rewind, Ch. F	X	
**REWG	Rewind, Ch. G	X	
**REWH	Rewind, Ch. H	X	
RFT	Right Half Indicators, Off Test		
RIA	Reset Indicators from the Accumulator		
RIL	Reset Indicators of the Left Half		
RIR	Reset Indicators of the Right Half		
RIS	Reset Indicators from Storage	X	X
RND	Round	X	
RNT	Right Half Indicators, On Test		
RPRA	Read Printer, Ch. A	X	
**RPRB	Read Printer, Ch. B	X	
RPRC	Read Printer, Ch. C	X	
**RPRD	Read Printer, Ch. D	X	
RPRE	Read Printer, Ch. E	X	
**RPRF	Read Printer, Ch. F	X	
**RPRG	Read Printer, Ch. G	X	
**RPRH	Read Printer, Ch. H	X	
RQL	Rotate MQ Left	X	
RTBA	Read Tape Binary, Ch. A	X	
RTBB	Read Tape Binary, Ch. B	X	
RTBC	Read Tape Binary, Ch. C	X	
RTBD	Read Tape Binary, Ch. D	X	
RTBE	Read Tape Binary, Ch. E	X	
RTBF	Read Tape Binary, Ch. F	X	
**RTBG	Read Tape Binary, Ch. G	X	
**RTBH	Read Tape Binary, Ch. H	X	
RTDA	Read Tape Decimal, Ch. A	X	
RTDB	Read Tape Decimal, Ch. B	X	
RTDC	Read Tape Decimal, Ch. C	X	
RTDD	Read Tape Decimal, Ch. D	X	
RTDE	Read Tape Decimal, Ch. E	X	
RTDF	Read Tape Decimal, Ch. F	X	
**RTDG	Read Tape Decimal, Ch. G	X	
**RTDH	Read Tape Decimal, Ch. H	X	
RUNA	Rewind and Unload, Channel A	X	
RUNB	Rewind and Unload, Channel B	X	
RUNC	Rewind and Unload, Channel C	X	
RUND	Rewind and Unload, Channel D	X	
RUNE	Rewind and Unload, Channel E	X	
RUNF	Rewind and Unload, Channel F	X	
*RUNG	Rewind and Unload, Channel G	X	
*RUNH	Rewind and Unload, Channel H	X	
SBM	Subtract Magnitude	X	X
SCHA	Store, Ch. A	X	X
SCHB	Store, Ch. B	X	X
SCHC	Store, Ch. C	X	X
SCHD	Store, Ch. D	X	X
SCHE	Store, Ch. E	X	X
SCHF	Store, Ch. F	X	X
**SCHG	Store, Ch. G	X	X
**SCHH	Store, Ch. H	X	X
SDHA	Set Density High, Channel A	X	
SDHB	Set Density High, Channel B	X	
SDHC	Set Density High, Channel C	X	
SDHD	Set Density High, Channel D	X	
SDHE	Set Density High, Channel E	X	
SDHF	Set Density High, Channel F	X	
*SDHG	Set Density High, Channel G	X	
*SDHH	Set Density High, Channel H	X	
SDLA	Set Density Low, Channel A	X	
SDLB	Set Density Low, Channel B	X	
SDLC	Set Density Low, Channel C	X	
SDLD	Set Density Low, Channel D	X	
SDLE	Set Density Low, Channel E	X	
SDLF	Set Density Low, Channel F	X	
*SDLG	Set Density Low, Channel G	X	
*SDLH	Set Density Low, Channel H	X	
SIL	Set Indicators of the Left Half		
SIR	Set Indicators of the Right Half		
SIX	Six		
SLF	Sense Lights Off	X	
SLN	Sense Lights On	X	
SLQ	Store Left Half MQ	X	X
SLT	Sense Light Test	X	
SLW	Store Logical Word	X	X
SPRA	Sense Printer, Ch. A	X	
**SPRB	Sense Printer, Ch. B	X	
SPRC	Sense Printer, Ch. C	X	
**SPRD	Sense Printer, Ch. D	X	
SPRE	Sense Printer, Ch. E	X	
**SPRF	Sense Printer, Ch. F	X	
**SPRG	Sense Printer, Ch. G	X	
**SPRH	Sense Printer, Ch. H	X	
SPTA	Sense Printer Test, Ch. A	X	
**SPTB	Sense Printer Test, Ch. B	X	
SPTC	Sense Printer Test, Ch. C	X	
**SPTD	Sense Printer Test, Ch. D	X	
SPTE	Sense Printer Test, Ch. E	X	
**SPTF	Sense Printer Test, Ch. F	X	
**SPTG	Sense Printer Test, Ch. G	X	

*709 system instruction only.
**7090 system instruction only.

CODE	COMMENT	INDEXABLE	IND. ADDRESS
**SPTH	Sense Printer Test, Ch. H	X	
SPUA	Sense Punch, Ch. A	X	
**SPUB	Sense Punch, Ch. B	X	
SPUC	Sense Punch, Ch. C	X	
**SPUD	Sense Punch, Ch. D	X	
SPUE	Sense Punch, Ch. E	X	
**SPUF	Sense Punch, Ch. F	X	
**SPUG	Sense Punch, Ch. G	X	
**SPUH	Sense Punch, Ch. H	X	
SSM	Set Sign Minus	X	
SSP	Set Sign Plus	X	
STA	Store Address	X	X
STD	Store Decrement	X	X
STI	Store Indicators	X	X
STL	Store Instruction Location Counter	X	X
STO	Store	X	X
STP	Store Prefix	X	X
STQ	Store MQ	X	X
STR	Store Location and Trap		
STT	Store Tag	X	X
STZ	Store Zero	X	X
SUB	Subtract	X	X
SVN	Seven		
SWT	Sense Switch Test	X	
SXA	Store Index in Address		
SXD	Store Index in Decrement		
TCH	Transfer in Channel		
TCNA	Transfer on Ch. A Not in Operation	X	X
TCNB	Transfer on Ch. B Not in Operation	X	X
TCNC	Transfer on Ch. C Not in Operation	X	X
TCND	Transfer on Ch. D Not in Operation	X	X
TCNE	Transfer on Ch. E Not in Operation	X	X
TCNF	Transfer on Ch. F Not in Operation	X	X
**TCNG	Transfer on Ch. G Not in Operation	X	X
**TCNH	Transfer on Ch. H Not in Operation	X	X
TCOA	Transfer on Ch. A in Operation	X	X
TCOB	Transfer on Ch. B in Operation	X	X
TCOC	Transfer on Ch. C in Operation	X	X
TCOD	Transfer on Ch. D in Operation	X	X
TCOE	Transfer on Ch. E in Operation	X	X
TCOF	Transfer on Ch. F in Operation	X	X
**TCOG	Transfer on Ch. G in Operation	X	X
**TCOH	Transfer on Ch. H in Operation	X	X
TEFA	Transfer on End of File, Ch. A	X	X
TEFB	Transfer on End of File, Ch. B	X	X
TEFC	Transfer on End of File, Ch. C	X	X
TEFD	Transfer on End of File, Ch. D	X	X
TEFE	Transfer on End of File, Ch. E	X	X
TEFF	Transfer on End of File, Ch. F	X	X
**TEFG	Transfer on End of File, Ch. G	X	X
**TEFH	Transfer on End of File, Ch. H	X	X
TIF	Transfer if Indicators Off	X	X
TIO	Transfer if Indicators On	X	X
TIX	Transfer on Index		
TLQ	Transfer on Low MQ	X	X
TMI	Transfer on Minus	X	X
TNO	Transfer on No Overflow	X	X
TNX	Transfer on No Index		
TNZ	Transfer on No Zero	X	X
TOV	Transfer on Overflow	X	X
TPL	Transfer on Plus	X	X
TQP	Transfer on MQ Plus	X	X
TRA	Transfer	X	X
TRCA	Transfer on Redun. Check, Ch. A	X	X
TRCB	Transfer on Redun. Check, Ch. B	X	X
TRCC	Transfer on Redun. Check, Ch. C	X	X
TRCD	Transfer on Redun. Check, Ch. D	X	X
TRCE	Transfer on Redun. Check, Ch. E	X	X
TRCF	Transfer on Redun. Check, Ch. F	X	X
**TRCG	Transfer on Redun. Check, Ch. G	X	X
**TRCH	Transfer on Redun. Check, Ch. H	X	X
TSX	Transfer and Set Index		
TTR	Trap Transfer	X	X
TXH	Transfer on Index High		
TXI	Transfer with Index Incremented		
TXL	Transfer on Index Low		
TZE	Transfer on Zero	X	X
UAM	Unnormalized Add Magnitude	X	X
UFA	Unnormalized Floating Add	X	X
UFM	Unnormalized Floating Multiply	X	X
UFS	Unnormalized Floating Subtract	X	X
USM	Unnormalized Subtract Magnitude	X	X
VDH	Variable Length Divide or Halt	X	
VDP	Variable Length Divide or Proceed	X	
VLM	Variable Length Multiply	X	
*WDR	Write Drum	X	
WEF	Write End of File	X	
WEFA	Write End of File, Ch. A	X	
WEFB	Write End of File, Ch. B	X	
WEFC	Write End of File, Ch. C	X	
WEFD	Write End of File, Ch. D	X	
WEFE	Write End of File, Ch. E	X	
WEFF	Write End of File, Ch. F	X	
**WEFG	Write End of File, Ch. G	X	
**WEFH	Write End of File, Ch. H	X	
WPBA	Write Printer Binary, Ch. A	X	
**WPBB	Write Printer Binary, Ch. B	X	
WPBC	Write Printer Binary, Ch. C	X	
**WPBD	Write Printer Binary, Ch. D	X	
WPBE	Write Printer Binary, Ch. E	X	
**WPBF	Write Printer Binary, Ch. F	X	
**WPBG	Write Printer Binary, Ch. G	X	
**WPBH	Write Printer Binary, Ch. H	X	
WPDA	Write Printer Decimal, Ch. A	X	
**WPDB	Write Printer Decimal, Ch. B	X	
WPDC	Write Printer Decimal, Ch. C	X	
**WPDD	Write Printer Decimal, Ch. D	X	
WPDE	Write Printer Decimal, Ch. E	X	
**WPDF	Write Printer Decimal, Ch. F	X	
**WPDG	Write Printer Decimal, Ch. G	X	
**WPDH	Write Printer Decimal, Ch. H	X	
WPUA	Write Punch, Ch. A	X	
**WPUB	Write Punch, Ch. B	X	
WPUC	Write Punch, Ch. C	X	
**WPUD	Write Punch, Ch. D	X	
WPUE	Write Punch, Ch. E	X	
**WPUF	Write Punch, Ch. F	X	
**WPUG	Write Punch, Ch. G	X	
**WPUH	Write Punch, Ch. H	X	
WRS	Write Select	X	
WTBA	Write Tape Binary, Ch. A	X	
WTBB	Write Tape Binary, Ch. B	X	
WTBC	Write Tape Binary, Ch. C	X	
WTBD	Write Tape Binary, Ch. D	X	
WTBE	Write Tape Binary, Ch. E	X	
WTBF	Write Tape Binary, Ch. F	X	
**WTBG	Write Tape Binary, Ch. G	X	
**WTBH	Write Tape Binary, Ch. H	X	
WTDA	Write Tape Decimal, Ch. A	X	
WTDB	Write Tape Decimal, Ch. B	X	
WTDC	Write Tape Decimal, Ch. C	X	
WTDD	Write Tape Decimal, Ch. D	X	
WTDE	Write Tape Decimal, Ch. E	X	
WTDF	Write Tape Decimal, Ch. F	X	
**WTDG	Write Tape Decimal, Ch. G	X	
**WTDH	Write Tape Decimal, Ch. H	X	
*WTV	Write Cathode Ray Tube	X	
XCA	Exchange Accumulator and MQ		
XCL	Exchange Logical Accumulator and MQ		
XEC	Execute	X	X
ZET	Storage Zero Test	X	X

*709 system instruction only.
**7090 system instruction only.

IBM 709-7090 Printed in U.S.A. D22-6508-2

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York