Reference Manual

IBM 7090 Data Processing System

Reference Manual

IBM 7090 Data Processing System

MINOR REVISION (March 1962)

This edition, A22-6528-4, is a minor revision of the preceding edition but does not obsolete Form A22-6528-3. The principal changes are the addition of Hypertape Drive information and added 7909 Data Channel information. Newsletters N22-0014 and N22-0019 are obsoleted by this edition.

Preface

This manual presents the operation and use of the IBM 7090 Data Processing System. Its purpose is twofold: (1) to provide a reference and guide for those familiar with the system; and (2) to serve as an instructional aid in the training of operators and programmers. The manual assumes that the reader is familiar with the IBM *709-7090 General Information Manual,* Form D22-6508.

The manual is divided into sections, each including related machine or functional operations. For example, information about any input-output device used with the systems is located in the "Input-Output Components" section. The sections are independent and need not be used in the order in which they appear.

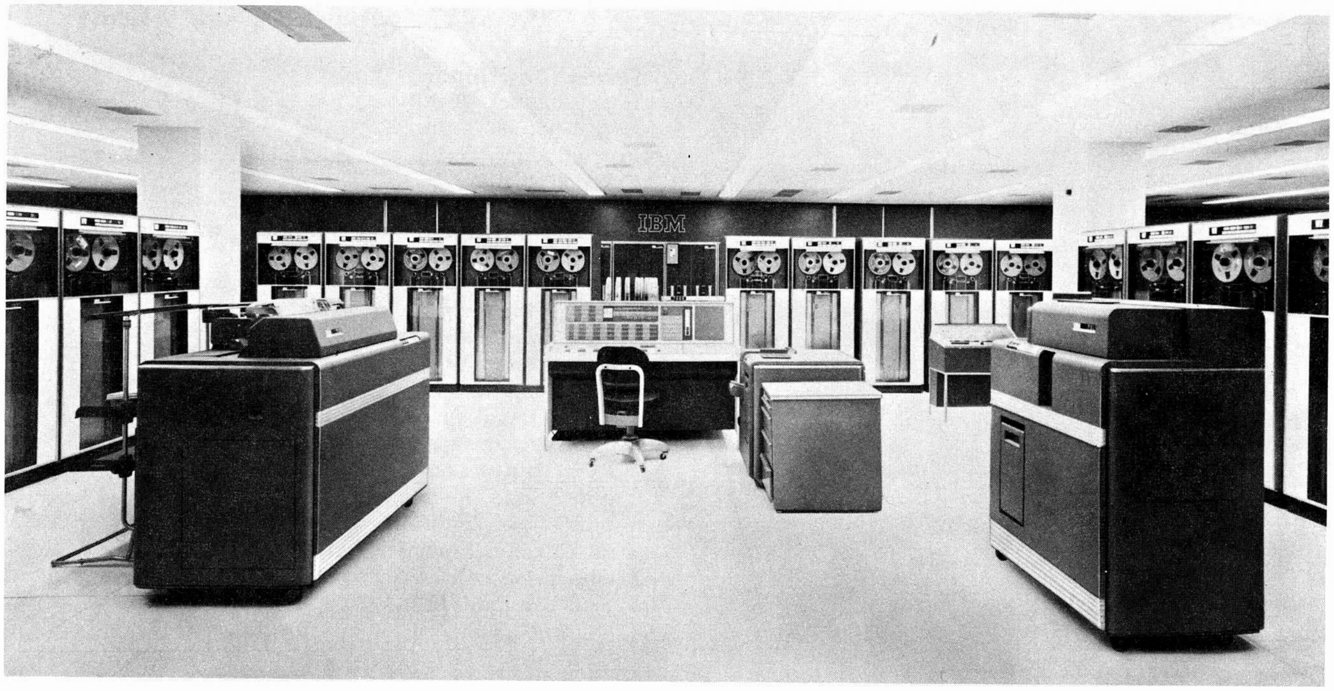
For each instruction, the manual gives: (1) pertinent facts about the instruction, in brief, (2) a detailed description with illustrations and examples, where needed, and (3) the indicators involved and execution time for each system. Sample programs are usually shown in SHARE coding rather than in machine language.

The first section of the manual is devoted to general information concerning the systems. This information is mainly designed as a review, so that the reader may understand the terms as the manual uses them.

A short introduction at the beginning of each section reviews general principles and explains the organization of the section.

Operation of the central processing unit console control, data channel console, and all input-output unit keys and lights together with wiring examples is described in the IBM *Operator's Guide for 7090 Data Processing System,* Form A22-6535. Also included in the operator's guide are information paths within the system and operational procedures.

Contents



IBM 7090 Data Processing System

IBM 7090 Data Processing System

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 demands machines that will compute, select, and correlate data at electronic speeds. The IBM 7090 Data Processing System can solve problems that cannot be solved in a lifetime of manual labor.

Core Storage

The computer uses, as its high-speed storage unit, an information holding device composed of magnetic cores. This core storage is subdivided into units called *words,* each word being identified by a number assigned to it. This identifying number is called an *address.*

The IBM 7302 Core Storage has a capacity of 32,768 words (Figure 1). Each word is comprised of 36 databits. When data are taken from or entered into a word location, *reference* must be made to this address.

The 36 bits of a word are used in two ways: as an *instruction* to the computer "ordering" a particular operation, or as the *operand* of an operation (data).

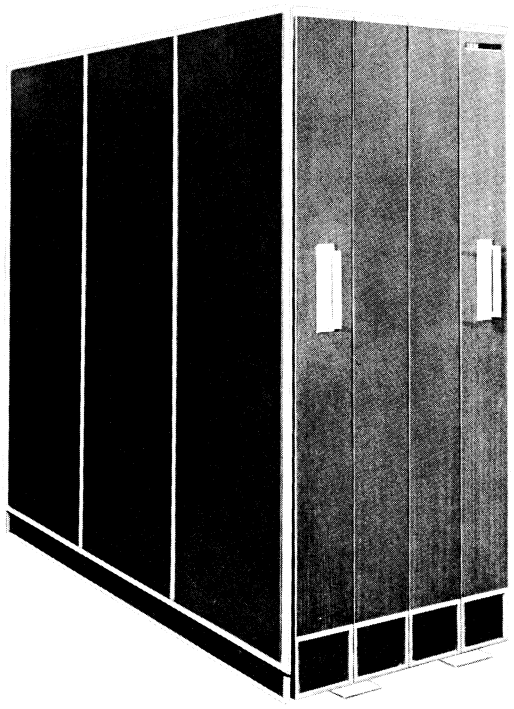


Figure 1. IBM 7302 Core Storage Unit

Stored Program

The computer does its work by executing many instructions at high speed. The set of instructions used in solving a problem form a *program* for the computer. Because the computer holds its instructions internally it is called a *stored program* system.

Normally, instructions are taken from sequentially ascending locations. However, the execution of instructions does not have to occur in this manner. It is possible, by using *control* or *transfer* instructions, to alter the sequential execution process and to indicate some other location as the next instruction to be executed. In this way, it is possible to repeat any instruction or block of instructions as often as desired.

The logical path followed by the program may be determined by a series of tests applied at points in the execution process, thus providing a stored program with the ability to control its own course of execution.

Fixed-Point Numbers

When a word contains a fixed-point number, the first of the 36 positions holds the algebraic sign of the number. A "0" signifies a positive number and a "1" signifies a negative number. The remaining 35 positions contain the magnitude of the number. Figure 2 shows locations in storage containing plus one and minus three. When fixed-point operations are used, the programmer must decide where the point is to be located. On the computer, the point which separates the integral part from the fraction part is termed a *binary point.*

```
+1  |0|00000000000000000000000000000000001|
     S 1                                35
-3  |1|00000000000000000000000000000000011|
     S 1                                35
```

Figure 2. Words Containing +1 and −3

Floating-Point Numbers

When the range of numbers anticipated during a calculation is either large or unpredictable, it becomes difficult to work with fixed-point instructions. An alternative set of floating-point instructions is available for such calculations. These instructions maintain the binary point automatically.

A floating-point decimal number X may be expressed as a signed proper fraction (N) multiplied by some integral power (n) of 10. The number is *normal* if the power of 10 (n) is chosen so that the decimal point is positioned to the left of the most significant digit of N. Examples:

X		N		10^n
—.010	=	—.10	×	10^{-1}
.140	=	.14	×	10^0
4.600	=	.46	×	10^1
88.000	=	.88	×	10^2

Likewise, a *floating-point binary number* (X) may be represented as a signed proper fraction (B) times some integral power (b) of 2. In the normalized case the binary point is positioned to the left of the most significant digit of B. Examples:

X(BINARY)		B(BINARY)		2^b(DECIMAL)
—.001	=	—.100	×	2^{-2}
.100	=	.100	×	2^0
1.100	=	.110	×	2^1
110.000	=	.110	×	2^3

In the computer a floating-point number is stored in a word as shown in Figure 3. The fraction is contained in bit positions 9 through 35. A floating-point

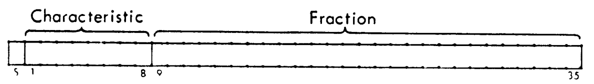


Figure 3. Floating-Point Word Format

number with a 1-bit in position 9 is said to be normal. The sign of the fraction is contained in the S position of the word. The characteristic 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. Since $128_{10} = 200_8$, the characteristic of a non-negative exponent always has a 1-bit in position 1, while the characteristic of a negative exponent always produces a 0-bit in position 1. A normal zero has no bits in both the characteristic and the fraction.

Instructions

Most computer instructions have an address part which is used to denote the location in core storage which is to be subjected to some arithmetic or logical operation. This address part or field always occupies bit positions 21 through 35 (Figure 4). The 15-bit address field is just large enough to hold the number 32,767, the highest core storage address. This number, expressed in binary, is simply 15 consecutive ones. On a computer having less than 32,768 positions of core storage, the higher positions of the address field are ignored by the computer.

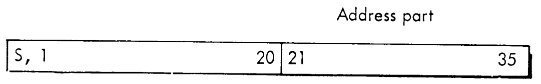


Figure 4. Address Part of the Instruction

The operation part of an instruction, in general, is not fixed in length but may vary from one instruction to another. Figure 5 shows the bit pattern for the ADD instruction specifying core location 0001.

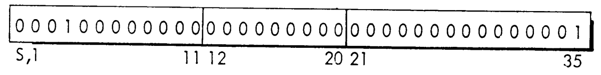


Figure 5. Add Instruction

Central Processing Unit

The central processing unit (Figure 6) consists of the IBM 7108 Instruction Processing Unit and the IBM 7109 Arithmetic Sequence Unit. All arithmetic and control functions are accomplished by these units. For ease of description, these units are called Central Processing Unit or CPU. This section describes the functions of major registers and counters within the CPU.

The *accumulator register* (AC) has a capacity of 37 bits and a sign position (Figure 7). In addition to the normal 36 positions, the AC has two extra positions, Q and P, which precede position 1. These two positions are provided for accumulator overflow. When the sum of 35-bit numbers is a 36-bit result, a carry occurs out of the high-order position (position 1) and is placed in the P position. Similarly, a carry from P is

Figure 6. Central Processing Unit

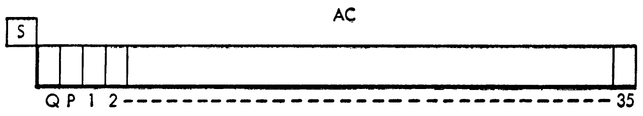


Figure 7. Accumulator Register

placed in Q. Carries from Q are lost. Whenever a carry from position 1 to position P occurs, as a result of a fixed-point arithmetic or shifting instruction, the overflow indicator is turned on. The status of the overflow indicator may be tested by two of the computer's conditional control instructions.

The *multiplier-quotient register* (MQ) has a capacity of 36 bits. It has a special function in multiplication and division operations. With regard to multiplication and some shift operations, the MQ may be considered as the right-hand extension of the AC register (Figure 8).

The *storage register* (SR) has a capacity of 36 bits and serves as a buffer between core storage and the CPU. It is used for both arithmetic and control functions. For this reason the contents of the SR are always destroyed before the execution of a new instruction. Therefore, it is not a normal object of concern to the programmer except when manually stepping through a program in order to locate program errors.

The *sense register* (SI) consists of 36 bits which may be addressed by any of a special group of sense indicator (SI) instructions. These operations treat the bits of the SI register as switches which may be logically manipulated and tested either singly or in groups. Figure 9 is a schematic of this register.

The *instruction counter* (IC), with a capacity of 15 bits (for 32,768 word core storage) determines the location from which the next instruction is to be taken. Normally the content of this register is the location or address of the current instruction, plus one. The highest location in core storage and location zero are treated as consecutive locations. During execution of test instructions the location counter (instruction counter) may be increased by 1, 2, or 3, resulting in a corresponding *skip* in the program. Similarly, during the execution of transfer type instructions, the contents of the IC may be replaced by the value specified by the transfer instruction.

The *instruction register* (IR) contains the operation part of the instruction-word. When the CPU is

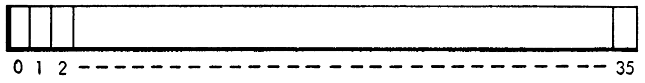


Figure 9. Sense Indicator Register

ready to accept another instruction, the word in core storage specified by the instruction counter is brought into the storage register. The operation part of the instruction is brought to the instruction register for interpretation and execution, while the remaining portion of the instruction is interpreted in the SR.

Three *index registers* (XR) are used in the computer. These registers are called A, B, and C or 1, 2, and 4. The latter terminology is convenient for the programmer because the numbers 1, 2, and 4 are the octal representation of the "addresses" of the three registers. These addresses are stipulated in a part of an instruction designated as the *tag field* and are normally referred to as "tags." This tag field always occupies bit positions 18, 19, and 20 of an instruction (Figure 10).

NOTE: all numbers (addresses) in the text are considered to be expressed in the octal system, unless otherwise stated.

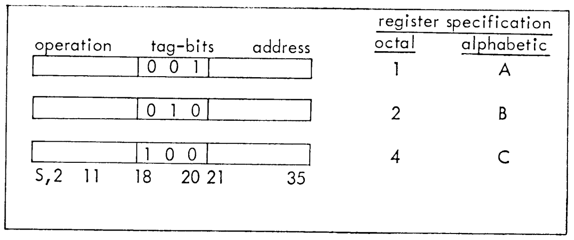


Figure 10. Index Register Tag Bits

Address Modification with Index Registers

One of the primary uses of index registers arises from their ability to modify instruction addresses. For this to occur, the instruction must specify the particular register or registers that are to take part in the modifying activity. This is done by the appropriate bit configuration in the tag field. The instruction is then executed as if its address field contained the stated address minus the contents of the index register. For example, assume that storage location 1000 contains the instruction ADD 2117 and that this instruction has a 2 in its tag field. If the contents of index register 2 are 117, the number stored in location 2000 (2117 minus 117) is added into the accumulator when the

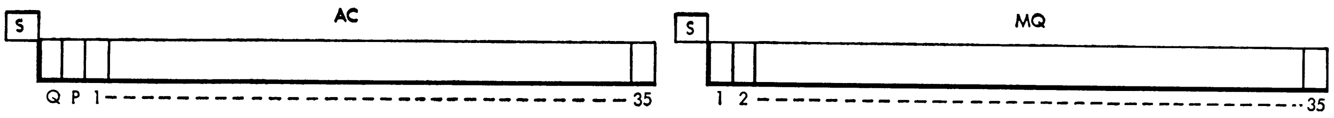


Figure 8. Accumulator and MQ Registers

ADD instruction is executed. However, location 1000 still contains the instruction ADD 2117 in its original form. Address 2000 is called the effective address, and the process is called effective address modification; that is, the address of the instruction is modified in the CPU for execution purposes but is unaltered in storage.

All computer instructions, when tagged, are subject to effective address modifications with the following exceptions:

1. Instructions which load, store, modify, or test the contents of an index register. These instructions use the tag field to specify the index register which is to be affected.
2. Convert instructions.
3. Those sense indicator instructions which combine the address and tag fields and use the entire right half of the instruction as a *mask*.

NOTE: The convert and sense indicator instructions referred to in items 2 and 3 above are defined in the section "Computer Instructions."

Multiple Tag

An instruction may refer to more than one index register by placing multiple 1's in the tag field (Figure 11). Thus, a tag of three specifies index registers 1 and 2. Care must be exercised when multiple tags are used. The use of multiple tags results in the "logical OR" of the contents of the specified index registers. For example, if a tag of three is given, the 15 positions of index register 1 are matched against the corresponding positions of index register 2. If *either* bit in a given position is a 1, the resulting logical sum will have a 1 in that position. If *both* positions are 0, the logical sum will have a 0 in that position. For example, assume index registers 2 and 4 contain 03204 (000011010000100) and 03061 (000011000110001), respectively. The instruction ADD 6521 with an index tag of 6 causes the number 03265 (000011010110101) to be subtracted from the address of the instruction and the effective address is therefore 03234.

Tag Field			
Binary	Octal	Index Registers	Specified
000	0	None	
001	1	A	1
010	2	B	2
011	3	A & B	1 & 2
100	4	C	4
101	5	A & C	1 & 4
110	6	B & C	2 & 4
111	7	A & B & C	1 & 2 & 4

Figure 11. Multiple Tags

Decrement Field

A group of instructions are used to test or alter the contents of an index register. The number used to test or alter an index register is contained in positions 3-17 of these instructions. These 15 bit positions are referred to as the decrement field (Figure 12).

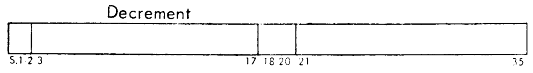


Figure 12. Decrement Field in a Word

Complement Arithmetic

When index registers are used for effective address modification, the contents of an index register are always subtracted from an instruction's address. Since neither the address of an instruction nor the contents of an index register is associated with any algebraic sign, it is not possible to accomplish effective address modification by addition in any direct manner. However, this may be accomplished by using complement arithmetic. The following definitions apply to this type of arithmetic:

1. The 1's complement of a number is defined as that number which results by replacing each 1 in a number with a 0 and each 0 with a 1. For example, given the binary number 101, the 1's complement would be 010. Also, the sum of a binary number and its 1's complement is a binary number composed of all ones (101 + 010 = 111).

2. The 2's complement of a binary number is defined as the 1's complement of a number increased by one. Thus, for the preceding example, the 2's complement of the binary number 101 would be 011. If the 2's *complement* of a number occupies an index register and is used to modify an address, the effective address is the *sum* of the index register contents and the address portion of the instruction. If the *true number* occupies the index register, the effective address is the *difference* between the index register contents and the address portion of the instruction.

Since both the contents of an index register and an instruction address are 15-bit numbers, all resulting carries to the sixteenth position will be lost.

Effective addresses are always formed in the computer by the addition of the 2's complement of the contents of the index register. This is an automatic feature. For example, if index register 4 contains the number 00005 and the instruction ADD 00015 with a tag of 4 is executed, the effective address is 00010:

2's complement of XR 4	111	111	111	111	011
ADD instruction address	000	000	000	001	101
Effective address (carry lost)	000	000	000	001	000

If index register 4 had contained the 2's complement of 00005, (that is, 77773) then the effective address would be 00022:

2's complement of XR 4	000	000	000	000	101
ADD instruction address	000	000	000	001	101
Effective address	000	000	000	010	010

Indirect Addressing

The concept of effective address modification is extended for a large group of instructions for which *indirect addressing* is provided. This extension is carried out in a very simple way. Just as index registers are "addressed" with a tag, indirect addressing is specified or "addressed" by a *flag* (1 bits in both positions 12 and 13 of the instruction). With both positions 12 and 13 of an instruction containing ones, the instruction is executed in the following way. An effective address is computed in the normal manner, by subtracting the contents of the specified index register, if one is specified, from the address part of the instruction. This is known as an *indirect effective* address. The calculator then examines the location specified by *this* indirect effective address and uses the tag and address parts of this word to compute a *direct effective* address. The instruction is then executed as if its address field had contained this direct effective address with no flag or tag. The following examples illustrate this process. Assume that the address part of location 00054 in core storage contains 00273. If the instruction ADD 00054 is executed, the contents of location 00054 will be added into the AC. However, if this same instruction had indirect addressing specified by 1 bits in both positions 12 and 13, the contents of location 00273 would be added into the AC. Now, assume further that index registers 1 and 2 contain 4 and 3, respectively, and that core storage location 00050 contains a 2 in its tag field and 00167 in its address part. If the instruction ADD 00054, with an index tag of 1 and indirect address flag specified, is executed, then the indirect effective address equals 00050 (address field of the ADD instruction minus the contents of XR 1). The direct effective address is 00164 (address part of location 00050 minus the contents of XR 2) and the contents of *this* location are added into the AC.

Indicators and Sense Devices

Three indicators are also contained in the CPU. These indicators are either on or off and can be tested by means of a test instruction peculiar to that indicator.

The *accumulator overflow* indicator is turned on whenever a 1 passes into or through position P from position 1 of the AC as a result of the execution of a fixed-point arithmetic or a shifting instruction. An example is a carry resulting from an algebraic addition. Either of the instructions TRANSFER ON OVERFLOW or TRANSFER ON NO OVERFLOW can be used to test the status of this indicator.

The *divide-check* indicator is turned on, in fixed-point division, if the magnitude of the number in the AC (dividend) is greater than or equal to the magnitude of the number in storage (divisor). In floating-point division a divide check occurs only when the divisor is zero or if the magnitude of the fraction of the dividend is greater than or equal to twice the magnitude of the fraction of the divisor. The divide-check indicator is tested by the DIVIDE CHECK TEST instruction.

The *input-output* check indicator (I-O check) is turned on by the attempted execution of an input-output instruction (copy, locate drum address, reset and load channel, or load channel) without selecting an input-output unit. Other conditions affecting the status of this indicator are discussed in the following pages. The I-O check indicator is tested by the INPUT-OUTPUT TEST instruction.

Transfer Trap Mode

The computer can be operated in a special transfer trap mode. The major use of this mode is in program testing. Operation in the trap mode permits the program to run at normal speed with interruptions of normal operation only at transfer points. At such points the location of the last sequential instruction is saved, and a transfer of control is made to a fixed location. Beginning at this fixed location, a special monitoring program may aid the programmer in control of his stored program even in the event of an incorrect transfer.

When the computer is operating in this mode, the location of each transfer instruction replaces the address part of location 0 prior to the instruction's execution. Unconditional transfers and conditional transfers for which the transfer conditions are met are not executed. Instead, control is transferred to location 1. One instruction, TRAP TRANSFER, is immune to the trapping mode. Its location is never stored in location 0, and control is always transferred to the location specified by the address of this instruction.

The instructions ENTER TRAPPING MODE and LEAVE TRAPPING MODE are used to enter or leave this special mode. Depression of the clear or reset keys on the console also causes the computer to exit from this mode.

Sense Switches

Located on the console are six sense switches, each of which may be turned on or off by the machine operator. The instruction SWT (switch test) is used to test the setting of any desired switch.

Sense Lights

Also located on the console are four sense lights. Any one of these lights may be turned on or off by the SLN and SLF instructions. Another instruction, SLT (sense light test) is used to test the status of any desired sense light.

Panel In-Out Switches

A group of 36 switches corresponding to the 36 positions of a word are provided on the console. A switch turned down corresponds to a 1, and one turned up corresponds to a 0. The instruction ENTER KEYS causes the number entered into these switches to replace the contents of the MQ register. A reset switch on the 7090 restores all input switches to the zero state.

Data Channel Operation

Data being transmitted between core storage and any input-output device must pass through a data channel. The operation of a data channel is initiated by the execution of two instructions in the central processing unit. Once started, the channel operates independently of the main program being executed by the CPU. A data channel has the responsibility for controlling the quantity and destination of all data transmitted between core storage and the input-output device. It also performs limited counting and testing operations concerned with the transmission of data.

The IBM 7909 Data Channel is also able to instruct and select an input-output device adapter, such as the IBM 7631 File Control or the IBM 7640 Hypertape Control.

Programs for a channel operation are stored in core storage just as are instructions executed by the CPU. To distinguish between CPU and data channel programs, data executed in the CPU are termed *instructions,* data executed by the data channel are termed *commands,* and data executed by the adapter are termed *orders.*

Although a channel, once started, operates asynchronously, the main program may exercise a large degree of supervisory control through instructions which test the status of a data channel. A single command may transmit a large block of words between core storage and an input-output device so that normally many instructions in the main program may be executed during the time taken to execute just one command in a data channel.

All transmission is in 36-bit word parallel fashion. Since the CPU and a data channel cannot take a storage reference cycle at the same time, the execution of an instruction in the main program may be delayed at least one computer cycle. Once such a delay occurs, all of the time needs of all data channels will be satisfied before the main program execution is resumed. Such delays are imposed automatically and do not interfere with the internal registers or calculations in the CPU. If the instruction being executed is not using core storage when a channel requires a storage cycle, normally no delay is occasioned. When several data channels require storage cycles at one time, the sequence of transmission is handled automatically.

A data channel controls all input-output units attached to it in much the same way. Because of the importance of magnetic tape, the relationship between a data channel and tape is discussed here. A description of the operation of the card reader, card punch, and printer, together with additional information on magnetic tape, will be found in the section "Input-Output Components."

Magnetic Tape and Data Channel (7607) Addresses

A maximum of ten tapes per channel may be used with the 7090 system. Like locations in core storage, each tape unit has an address. Each data channel also has an address. The combination of the two addresses will then specify a particular tape unit attached to a particular data channel.

To start a tape unit for reading or writing, a SELECT instruction must be executed in the main program. The instruction READ SELECT prepares the tape for a reading operation and the instruction WRITE SELECT prepares it for writing. The joint address of a tape unit and data channel occupies the address part (positions 21-35) of the select instruction. If the address field of this instruction is viewed as a five-digit octal number, then the three low-order digits specify the tape unit, and the fourth and fifth digits the data channel (Figure 13).

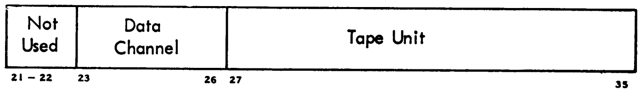


Figure 13. Select Instruction Address Field

The specific numerical addressing system used by the tapes is as follows:

1. Data channels A through H are identified by the octal numbers 1 through 10 and occupy positions 23-26 of the instruction.
2. In the BCD mode, tape units 1-10 are identified by the octal numbers 201-212, occupying positions 27-35 of the instruction.

3. In the binary mode, the tape units are identified by the octal numbers 221-232 which occupy positions 27-35 of the instruction.

Examples: If the instruction READ SELECT 1201 is executed, tape unit 1 attached to data channel A will be selected and started for a reading operation in the BCD mode. If the instruction WRITE SELECT 3223 is given, tape unit 3 attached to data channel C will be selected and started for a write operation in the binary mode.

Data Channel Registers (7607)

Once the select instruction has been executed, the operation of the tape and transmission of data between core storage and tape are under control of the data channel. There are four registers in the channel which control its operation. These registers are similar in function to the control registers in the CPU.

The first command of a data channel program must be sent to the channel by a RESET AND LOAD CHANNEL instruction from the main program. The address part of this instruction specifies the location in core storage containing the data channel command. When this instruction is executed, the contents of the location in core storage (specified by the RESET AND LOAD CHANNEL instruction) are sent to the data channel control registers.

There is a separate RESET AND LOAD CHANNEL instruction for each data channel. Where select instructions specify the appropriate data channel through usage of part of their address field, the address field of the reset and load channel is fully occupied by the storage address of the command. Thus, the distinction between data channels is made in the operation part of the instruction.

Word Count Register. The contents of positions 3-17 of the data channel command are loaded into this register (Figure 14). This register specifies the number of words to be transmitted between core storage and the input-output unit. As each word is entered or taken from core storage, the contents of the word register are reduced by one.

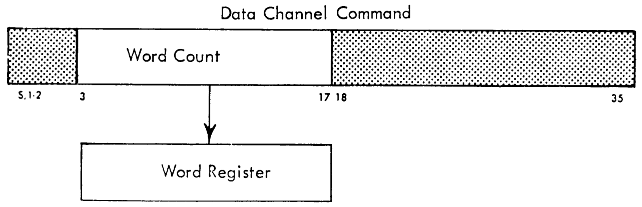


Figure 14. Data Channel Word Count Register

Channel Address Register. The contents of positions 21-35 of the data channel command are loaded into this register (Figure 15). The register specifies the location in core storage from which the data are to be taken during writing or to be entered into during reading. The contents of this register are increased by one after each word transmission to or from core storage. Thus, the address register directs the transmission of data into or from consecutive locations in core storage.

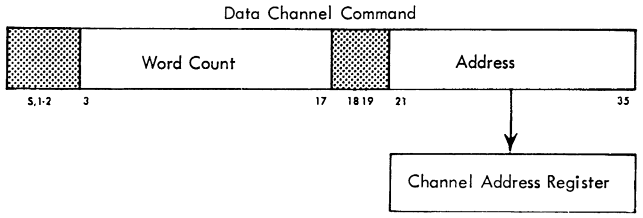


Figure 15. Data Channel Address Register

Location Register. This register is similar to the instruction counter in the CPU. The location register contains the location of the current data channel command, plus one. Thus, data channel commands are taken normally from sequential locations in core storage. Just as control or transfer instructions alter the contents of the CPU's instruction counter, transfer commands change the contents of the location register in a data channel.

The size of these registers (word register, channel address register, and location register) is set at 15-bit length. Each register has a capacity large enough to hold the address of the last location in core storage. When a 15-bit command field is loaded into a channel register, the leftmost bits which exceed the capacity of the register are ignored. When the contents of a channel register are stored in a 15-bit field in core storage, the leftmost bits corresponding to the absent bits of the register are set to zeros.

With reference to blocks of consecutively located commands or data words, the highest location in core storage and location zero are treated as consecutive locations.

Operation Register. This register is similar to the instruction register in the CPU. The contents of positions S, 1, 2, and 19 of a data channel command are loaded into this register (Figure 16). Bit positions S, 1, and 2 provide for eight possible data channel operations. Each of the eight commands may accomplish either reading or writing. Position 19 is used only for reading operations; it has no effect on writing operations. When this position contains a 1, all functions proceed normally except that no transmission

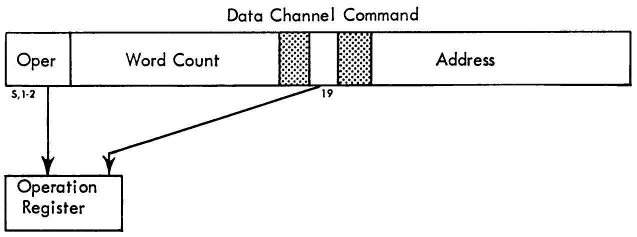


Figure 16. Data Channel Operation Register

of data to core storage occurs. Thus, a command with position 19 containing a 1 may be used to skip over a number of words (determined by the word count) while reading from an input device. When position 19 contains a 1, the data channel involved is said to be operating in the non-transmitting mode.

Data Register. This 36-bit register serves as a buffer between core storage and an input-output device.

Data Channel Register Example. An example is shown in Figure 17. If core storage location 1546 contains the data channel command, represented here as an octal number, 000124002117, and the instruction reset and load channel A with an address of 1546 is executed, then zeros will be placed in the operation register, 00124 will be placed in the word register, 2117 will be placed in the channel address register, and the location register will contain 1547.

Once a tape unit has been started by a select instruction, the data channel must receive its first command within a definite time period. Thus, the reset and load channel must be executed by the main program within this allotted time period following the select order. Specific tape timings are given in the magnetic tape section of "Input-Output Components."

If the reset and load channel is not executed within the allotted time period, the tape unit is logically disconnected from the calculator and no word transmission will occur. If a reset and load channel is given at any time when an input-output device is not logically connected to the data channel, the instruction is executed in the CPU but a special indicator, called the input-output check indicator, is turned on. The status of this indicator may be tested by the stored program.

Exact description, together with examples, of the data channel commands is found in the "Computer Instruction" section of this manual.

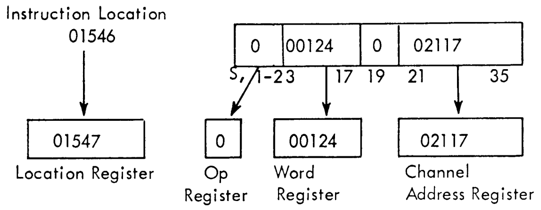


Figure 17. Data Channel Register Example

Data Channel Indicators (7607)

Each data channel has four indicators that may be turned on during tape operations. These indicators may be tested for the ON condition. When tested, an indicator that is on is turned off.

Tape Check Indicator. The tape check indicator may be turned on at any time during a tape read or write operation. When on, the indicator signals that an error has occurred in a read or write operation. The indicator will not be turned on unless the channel is logically connected to the CPU. For example, if the tape has been logically disconnected from the computer and is spacing to the next end-of-record gap, the indicator will not be turned on if an error is sensed during this period.

Beginning-of-Tape Indicator. Small strips of adhesive aluminum material are placed a few feet from each end of the tape. These strips are used to indicate the beginning of tape (load point) and the physical end of usable tape. Any instruction which backspaces the tape to its load point or attempts to backspace the tape beyond its load point turns on the beginning-of-tape indicator in the data channel to which the tape is attached, indicating that the backspace instruction was not logically completed.

End-of-Tape Indicator. When the strip marking the end of tape is reached during writing, the end-of-tape indicator in the data channel to which the tape is attached is turned on. No interruption in the writing process occurs so that the writing operation may be completed even though the end-of-tape strip has been passed over. However, if the status of the indicator is ignored and writing continues, the tape may eventually be pulled from its reel. This indicator is never turned on during a read operation.

End-of-File Indicator. The end-of-file indicator in a channel will be turned on any time an end of file (tape mark) is encountered during a reading operation. An end of file may be written on a tape at any time by the WRITE END OF FILE (WEF) instruction. The indicator is not turned on when an end of file is written.

The indicators for tape check, end of file, beginning and end of tape may be turned on by any of the tape units attached to a given data channel. To make a meaningful test of the indicators, therefore, the stored program should know which of the attached tape units had the possibility of turning on an indicator. The data channel's end-of-file indicator may also be turned on by a card reader attached to that channel.

In addition to the instructions which test the status of the indicators defined above, each data channel has a set of instructions which facilitate the synchronous operation of a stored program and its associated input-output activity. Two of these instructions test whether or not a channel operation is still in process. By the execution of another instruction it is possible to obtain, at any time, the contents of the data channel's operation register, channel address register, and location register.

When an end-of-file is sensed during reading, the turning on of the channel's end-of-file indicator logically disconnects the input-output device from the data channel. The execution of the channel command is terminated immediately, even if it has not been completed. The internal registers of the data channel are not reset. By obtaining the status of the location register and the address register, the main program may always ascertain the point at which the end-of-file condition interrupted the input-output operation.

Data Channel Trap

This feature allows the data channel to signal or interrupt processing by trapping the computer program. The trap may be initiated by: (1) the completion of a channel command, (2) an end of file, or (3) a redundancy tape check. These conditions are called *channel signals.*

Two instructions, ENABLE (ENB) and RESTORE CHANNEL TRAPS (RCT), are used with this feature and are described in the "Computer Instructions" section.

A *trap indicator* on the operator's console indicates when a trap occurs. The execution of an enable or restore channel trap instruction will turn the indicator on. The execution of any trap will turn the indicator off. With the indicator off, traps are said to be *inhibited.*

A data channel is *enabled* by use of the enable instruction. This instruction conditions the channel so that a channel signal may be combined with it, when and if it occurs. When a channel is enabled for a particular channel signal, all other channel signals will not be used for trapping. Thus, the channel is said to be *disabled* with regard to these other channel signals. A logic flow chart of circuits involved with one data channel is shown in Figure 18. Note that the trap control indicator must be on for a trap to occur.

Data channels may be individually or collectively prevented from causing traps until the program is able to handle them. In this event the channel signal is saved until the program allows it to become effective.

When a trap occurs, the contents of the instruction counter (IC) are stored and the next instruction is taken from a fixed location as follows:

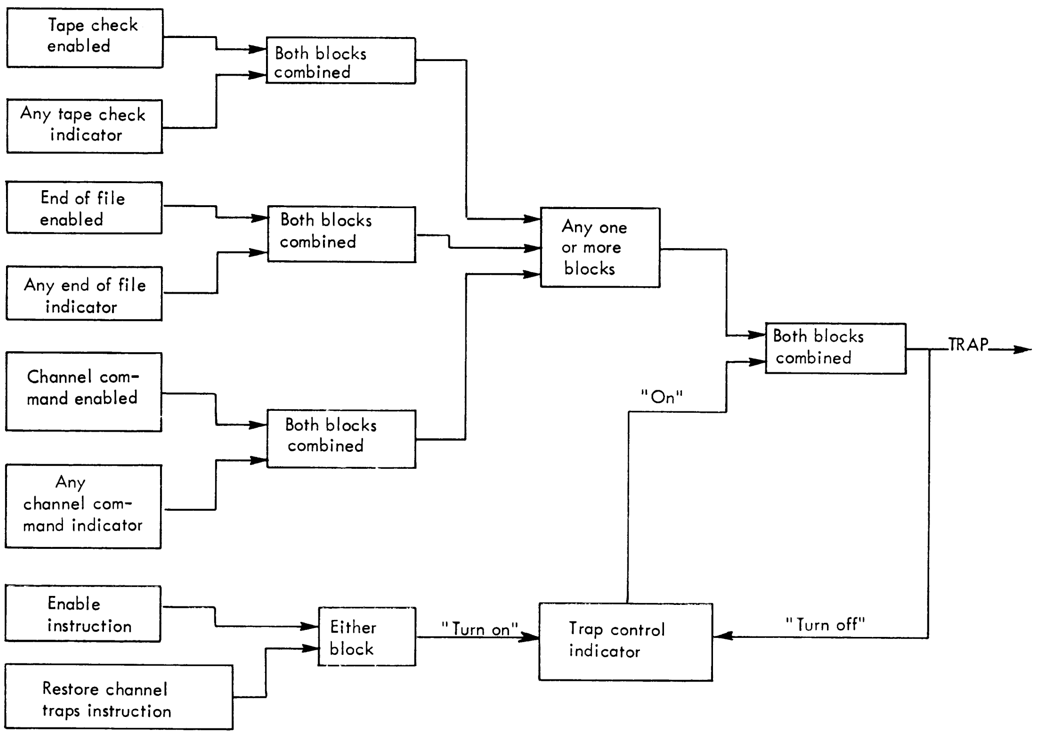


Figure 18. Logic Flow of Data Channel Trapping

CHANNEL	STORE THE IC AT	NEXT INSTRUCTION FROM
A	0012	0013
B	0014	0015
C	0016	0017
D	0020	0021
E	0022	0023
F	0024	0025
G	0026	0027
H	0030	0031

The first instruction after execution of a trap should be an unconditional transfer. If these instructions, located at the odd locations 0013-0031, are not provided by the programmer and do not alter the contents of the instruction counter, the program resumes from the point at which the trap occurred (after executing the instruction at the odd location). A trapping signal occurs under the conditions listed below:

1. If an IOCT, IORT, or IOST is used and no load channel instruction is waiting in the main program upon completion of the command. A trap resulting from this condition will cause the decrement of the location in which the contents of the instruction counter are stored to be cleared and a 1 to be placed in position 17.
2. Whenever the end-of-file indicator is turned on. A trap resulting from this condition will cause the decrement of the location in which the contents of the instruction counter are stored to be cleared and a 1 to be placed in position 15.
3. Whenever a redundancy check occurs. With this type of trap, the decrement of the location in which the instruction counter is stored is cleared and a 1 is placed in position 16.

If a trapping signal is generated while a channel is disabled or inhibited, the trap request is remembered until the channel is enabled or restored.

The instruction following an enable, restore channel trap, or execute instruction will always be executed before another trap is processed. Furthermore, traps are prevented from occurring between a read or write select and the following (normally a reset and load) instruction.

Execution of certain instructions, while a channel is disabled, will cause the remembered trap to be lost. This type of trap and the effect it produces are as follows:

TYPE	EFFECT
Channel command	Read or write selection of the corresponding channel
End-of-file	Execution of a transfer on end-of-file for that channel
Tape check	Execution of a transfer on tape-check for that channel

A TEF or TRC instruction, to a disabled channel, will reset the EOF or tape check indicator and will be executed properly.

The end-of-file and tape-check indicators are turned off whenever a trap results from an indicator's being on. If the tape-check indicator is on for an enabled channel (even though traps are inhibited), the channel is immediately disconnected.

When a channel is enabled for a tape-check or end-of-file condition, a transfer-on-redundancy-check or end-of-file instruction addressing that channel will be treated as a no-operation instruction.

If a trap is called for, subsequent to the execution of certain halt instructions, the following procedure occurs:

1. *Halt and transfer.* The trap is performed and the CPU resumes execution of instructions. At the time the trap occurs, the instruction counter contains the location of the HTR instruction.
2. *Halt and proceed.* This is the same procedure as HTR except that the instruction counter contains the location of the HPR instruction plus one.
3. *Divide or halt.* The computer is restarted as with the HTR, except that the instruction counter contains the location of the divide instruction plus one.

A trap normally occurs at the completion of the instruction being executed. For example, if a trap is called for while execution of a load-channel instruction is being delayed (command is incomplete), the trap does not occur until the load-channel instruction has been completed. If a trap is called for while the CPU is in manual status, no trap occurs until the CPU is returned to automatic status and the start key is depressed.

Programming Techniques

For programs utilizing data channel trap, certain characteristics of the feature make some programming techniques dangerous. If these techniques are used in a program operating with data channel trap enabled, random failures may occur which appear to be machine malfunction. Usually these failures are not reproducible, owing to the random occurrence of the I-O interrupts. Some techniques to be avoided are:

1. Execution of TRCX or TEFX instructions, when the I-O interrupts are disabled, turn off the corresponding trap indicator if it is on, causing a waiting trap to be lost. If the indicator was on for either of these conditions the instruction transfers correctly. Care must be taken when testing these conditions, both in and out

of the trap mode. A separate error checking routine should be used for operating in each mode, since conditions are different in both modes of operation.

2. Execution of TCOX* does not guarantee that a channel is not busy upon release. If interruption occurs because of a condition sensed at I-time of the TCOX, the channel is disconnected, causing the TCOX to advance the instruction counter to the next location following the TCOX. The interrupt now occurs, as the instruction is complete. If the interrupt routine re-initiates I-O activity on the channel (to maintain highest possible I-O speed), the channel is busy upon re-entry, but the TCOX has been bypassed. If an I-O operation follows the TCOX, its execution may cause the trap normally resulting from the previous channel activity to be dropped, also possibly causing the program to hang up in a waiting loop. It is suggested that a storage flag be kept for each channel to indicate channel activity. This flag may be tested in a:

```
    ZET         FLAG
    TRA         *-1
```

Loop to act as pseudo-TCOX* delay. The flag must be set to zero by the trap routine and set to non-zero by any routine which executes a select on the channel.

3. No data cell common to both trap and non-trap routines should be changed by the non-trap routine without disabling the interrupt system, unless the cell may be modified by one and only one instruction. This is illustrated by the following program which violates the above rule:

```
           . . . .save AC
TRAP       CLA BETA
           ADD ONE
           STO BETA
           . . . .restore AC
           . . . .
NONTRP     CLA BETA
           SUB ONE
           STO BETA
```

If a trap occurs between NONTRP and NONTRP +2, the routine at trap will save the AC, add one to BETA, and store the augmented value back in BETA. Upon reloading the AC and re-entering the NONTRP routine, the old contents of BETA are stored over the newly augmented value. This could be prevented by disabling the I-O traps before NONTRP and re-enabling them after NONTRP +2.

4. If the I-O program is checking for noise records in the end of record gap, the redundancy trap should not be used to detect noise. Redundancy trapping could occur while reading the first or second word from tape, immediately disconnecting the channel from the tape. A SCHX instruction to investigate the length of this redundant record would show that the record is only one or two words long, classing it as a noise record. This is an error and may be prevented by always testing for redundancy during the end of operation or end-of-file trap routine with a normal TRCX instruction.

5. Care must be taken when writing subroutines which may be entered by both trap and non-trap routines. Entry at trap time to a routine which has been interrupted by the trap can give destructive results.

6. Trap time subroutines which issue I-O selects should insure that the selected channel is dormant before giving the select. This may be difficult without destroying or ignoring existing trap signals on the selected channel.

For more information on trapping, see "Interrupt."

External Signal

A standard feature of the computer system is its ability to accept a signal from an external source. This signal is stored and will cause the computer to execute a trapping operation as soon as possible. Normally, upon receipt of an external signal, the instruction being executed is completed and the location of the next instruction is stored in the address portion of core location 0003. The computer then takes its next instruction from location 0004. If, however, the current instruction is a floating point trap instruction which causes overflow or underflow, the floating point trap will be executed before the external signal trap is allowed to occur. The external signal trap is completely independent of data channel traps. Significant implications of this are:

1. If the computer is halted, the trap will not occur until the start key is depressed and the halt operation is completed.
2. It is not possible to disable or inhibit an external signal trap.
3. The external signal trap is unconditional in that it may occur between an input-output select instruction and the following instruction, a restore channel traps or enable instruction and the following instruction, or an execute instruction and the instruction to be executed. If a trap occurs at one of these times and the external signal trap subroutine is then interrupted by a data channel trap, timing limitations may be exceeded and/or the program may return to the incorrect re-entry location in the main program.

Computer Instructions

This section defines all computer instructions and describes their execution, indicators that may be affected, and timing.

A diagram representing the format of the instruction is given for each instruction. Preceding this diagram is the alphabetic code which identifies the instruction. The official name of the instruction is also given (Figure 19).

CLA — Clear and Add

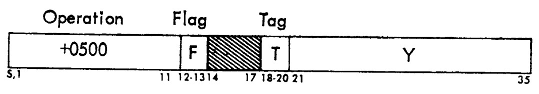


Figure 19. Sample Format of Instructions

The numerical operation code is given in the octal number system. This can be easily converted to the binary system for reference to the bit pattern interpreted by the computer. The numbers appearing beneath the diagram indicate the bit positions of the computer-word that are concerned with this particular instruction.

The symbol "Y" appearing in the diagram denotes the address part of the instruction. Y may stand for the address of a word in core storage, the length of a shift, or the address of an input-output unit. For some index transmission instructions, Y may also represent a number which is to be loaded either in true or complement form into an index register.

For some instructions, positions 21-35 are used to contain part of the operation code. The appearance of octal numbers instead of Y in the address field will distinguish this type of instruction from others. In all cases, the full operation code is shown by its octal representation.

Those instructions for which indirect addressing may be specified will have the symbol F (flag) appearing in positions 12 and 13 of the instruction diagram (Figure 19). This symbol represents 1 bits in both positions 12 and 13 of the instruction. The description of those operations which can have indirect addressing will be defined in terms of direct addressing.

Similarly, for instructions that are subject to effective address modification by an index register, the diagram has the symbol T in the tag field of the instruction. This T is also used to specify any index register to be changed, stored, or tested. The description accompanying an instruction defines the manner in which it is executed when its tag is zero.

The shaded area in the instruction diagrams represent fields that are not used in that instruction.

The symbols D, C, and R are used to denote the decrement, count, and right-half word fields of instructions which use these fields. Each of these fields is interpreted only by certain classes of instructions. If such a field is interpreted by an instruction, the bit positions used by the field will be shown in the instruction diagram. If an instruction has a D or R part, neither indirect addressing nor effective address modification is ever possible.

Descriptions of the instructions use the following special terms and definitions:

1. C (Y) denotes the contents of location Y, where Y refers to some location in storage. Similarly, C (AC), C (MQ), C (SR) and C (SI) denote the contents of the accumulator, multiplier-quotient, storage and sense indicator registers, respectively. In addition, subscripts refer to individual bit positions of a register. For example, $C\,(MQ)_{S,\,1\text{-}17}$ is read "the contents of positions S, 1 through 17 of the MQ." When subscripts are not used with this notation, the entire register is implied. For example, C (AC) denotes the contents of positions S,Q,P, 1-35, inclusive.

2. With input-output operations, DC denotes data channel, LR denotes a DC location register, AR denotes a DC address register, and WR denotes a DC word count register.

3. When a register or part of a register is *cleared,* the cleared part is reset to zeros.

4. The *negative* of a number is the number with its sign reversed.

5. The magnitude of a number is the number with its sign made positive. (A zero in position S corresponds to a positive sign.)

6. When the word "store" is used in the title of an instruction, the transmission of a word or part of a word *from* some special register (e.g., the AC, MQ, SI or an index register) *to* some location in core storage is always implied.

7. When the word "load" is used in the title of an instruction, the transmission of a word or part of a word *from* some location in core storage *to* some special register (e.g., the MQ, SI, or DC registers, but not the AC) is always implied.

8. When the word "place" is used in the title of an instruction, the AC is always one of the agents.

9. All logical operations interpret the sign position (S) of Y as a numerical binary bit corresponding to position P of the AC or position 0 of the SI. The S position of the AC is either ignored or cleared by logical operations.

10. In the three-letter alphabetic code:
    a. The letter Q designates the MQ register.
    b. The letter X in the second or third position designates an index register.
    c. The first letter of all transfer instructions is a T.

In the following instruction descriptions, an instruction format is shown for each instruction. Under the "Indicator" heading, only those indicators that may alter the course of a program through test instructions or by trapping are noted. Under the "Execution" section, when instructions are similar, only the differences are noted and a statement (e.g., "Same as ADD procedure") will mean that the operations are alike except for the differences noted. Instruction flow charts are used with many instructions to aid in presenting the data flow.

Note again that all addresses and numbers, unless otherwise specified, are given in the octal number system.

Instruction Timing

All instructions are listed in the appendix in alphabetic and numerical sequence. Timing is noted in cycles with modification type, if any. The 7090 cycle is 2.18 microseconds. If an instruction is subject to address modification through indexing and/or indirect addressing the facts will be noted by a T or F, respectively. With indirect addressing, the execution time is increased one cycle. The modification types are:

Type 1 Instructions. Multiply instructions are executed in two cycles if the number brought from storage contains zeros in positions 1 to 35. If the number brought from storage is not all zeros, execution time is a function of the number of sequential zero bits in the multiplier.

Type 2 Instructions. The execution time of these instructions is determined by the count field (C) specified in positions 10 through 17. The maximum number of cycles for a given value is $C/3 + 3$. Any remainder should be discarded.

Type 3 Instructions. FAD, FAM, FSB, and FSM will be executed in 6 cycles if the difference in characteristics is greater than 63 or if the extent of shift is less than 10 places during the adjustment of characteristics (step 5); also if the extent of shift is less than four places when normalizing (step 9b).

Type 4 Instructions. UFA, UAM, UFS and USM will be executed in five cycles if the difference in characteristics is greater than 63 or if the extent of shift is less than 10 places in step 5.

Type 5 Instructions. The execution of a FDH or FDP instruction requires only three cycles if the fraction of the dividend is zero.

Type 6 Instructions. The execution of a convert instruction is increased by one cycle for each storage reference specified by the count field in positions 10 through 17 of the instruction.

Type 7 Instructions. The instruction will be executed in two cycles if the extent of shift is 16 places or less. Each additional 12 shifts, or portion thereof, require another cycle.

Type 8 Instructions. The execution of these instructions may be delayed an indefinite length of time after interpretation, depending on the status of the I-O unit. For example, if multiple select instructions are given for the same data channel, the second select will be delayed if both selects are of the data-select type of operation.

All variable cycle instructions that have a precise minimum, average and maximum number of machine cycles are shown in Table I.

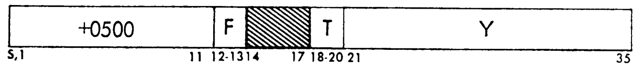
Table I. Variable Cycle Instructions

	MACHINE CYCLES		
INSTRUCTIONS	AVERAGE	MIN.	MAX.
MPY, MPR	11.6	2	14
DVH, DVP	14	3	14
FMP, UFM	11	2	13
FDH, FDP	13	3	13
FAD, FAM, FSB, FSM	6.4	6	15
UAM, USM	—	5	11
UFA, UFS	—	5	10
ALS, ARS, RQL	—	2	4
LLS, LRS, LGL, LGR	—	2	7
CAD, CAQ, CVR	—	2	8
VDH, VDP, VMP	—	2	14

Instructions with a count whose value is larger than that implied by the size of the arithmetic registers may exceed the times shown. Average multiply times are derived assuming a random distribution of ones and zeros. In floating-point, a normalized operand is assumed. In determining the average floating-point add speed, a number of representative programs were traced. The time shown is based on an analysis of several million operands.

Fixed Point Operation

CLA — Clear and Add



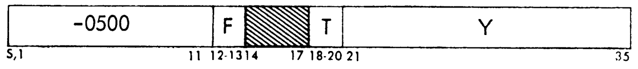
Description. The C(AC)$_{S,\,1\text{-}35}$ are replaced with the C(Y). Positions P and Q of the AC are set to zero. The C(Y) remain unchanged.

Indicators. None.

Timing: 2 cycles

Execution. The C(Y) are brought to the SR. C(SR)$_{1\text{-}35}$ is taken to the adders, the adders to AC$_{(1\text{-}35)}$ and the SR(S) to AC(S).

CAL — Clear and Add Logical Word



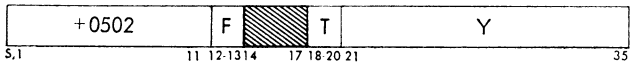
Description. The C(Y) replace the C(AC)$_{P,1\text{-}35}$. The sign of Y appears in position P of the AC. Positions S and Q of the AC are set to zero. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. The SR (S) goes to adder position P. The rest of the operation is the same as for CLA.

CLS — Clear and Subtract



Description. The negative of C(Y) replaces the C(AC)$_{S,1\text{-}35}$. Positions P and Q of the AC are set to zero. The C(Y) are unchanged.

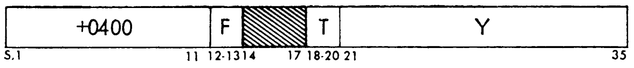
Indicators. None.

Timing: 2 cycles

Execution. (1) Invert sign of Y as it is entered into the SR. (2) Same as CLA.

The logic flow diagram for both the CLA and CLS instructions is shown in Figure 20.

ADD — Add



Description. The C(Y) are algebraically added to the C(AC). The resulting sum is placed in the AC.

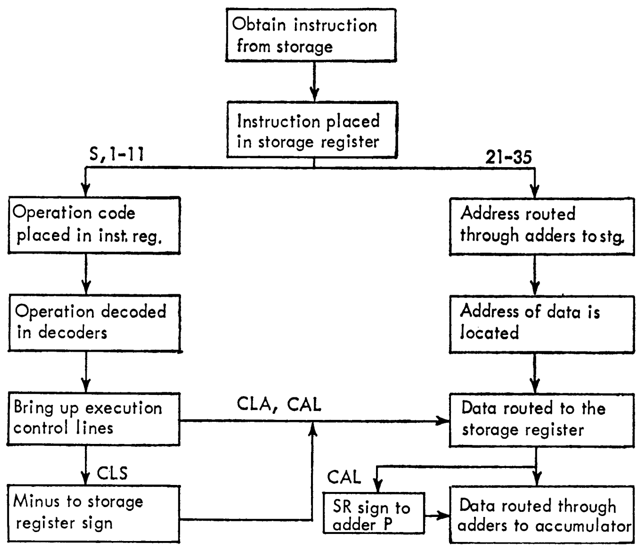

Obtain instruction from storage
Instruction placed in storage register
S,1-11
21-35
Operation code placed in inst. reg.
Address routed through adders to stg.
Operation decoded in decoders
Address of data is located
Bring up execution control lines
CLA, CAL
Data routed to the storage register
CLS
CAL
Minus to storage register sign
SR sign to adder P
Data routed through adders to accumulator

Figure 20. CLA and CLS Flow Chart

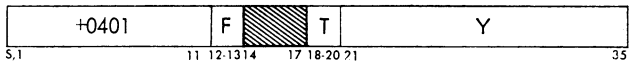
The C(Y) are unchanged. Numbers of the same magnitude but different signs give a resultant sign the same as the sign of the original AC.

Indicators. AC overflow.

Timing: 2 cycles

Execution. The C(Y) are taken to the SR and then to the adders. With signs alike, the true AC (Q-35) is also taken to the adders, and the sum returned to the AC. With signs unlike, the complement of the AC (Q-35) is taken to the adders; any Q carry is taken to adder 35 and is remembered. The resultant sum in the adders is then taken back to the AC. If the signs were unlike and there was no Q carry, the complement of the AC (Q-35) is again taken to the adders and then back to the AC. With a Q carry, reverse the AC sign (Figure 21).

ADM — Add Magnitude



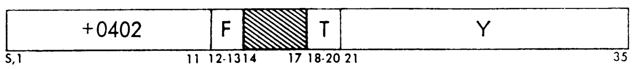
Description. The magnitude of the C(Y) is added to the C(AC). The resulting sum is placed in the AC. The C(Y) are unchanged. The sign of Y is ignored and Y is treated as a positive number. With a minus AC sign, a subtractive process will occur.

Indicators. AC overflow.

Timing: 2 cycles

Execution. (1) SR(S) is forced plus. (2) Procedure is the same as for ADD.

SUB — Subtract



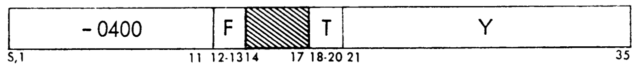
Description. The C(Y) are algebraically subtracted from the C(AC). The difference replaces the C(AC). The C(Y) are unchanged.

Indicators. AC overflow.

Timing: 2 cycles

Execution. (1) Sign of SR is reversed. (2) Same as ADD procedure.

SBM — Subtract Magnitude



Description. The magnitude of the C(Y) is subtracted from the C(AC). The difference is placed in the C(AC). The sign of Y is ignored and the C(Y) are treated as a negative number. The C(Y) are unchanged. If the sign of the AC is minus, an ADD will occur.

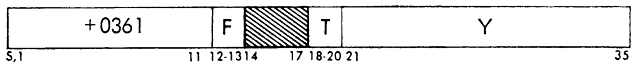
Indicators. AC overflow.

Timing: 2 cycles

Execution. (1) SR(S) is forced minus. (2) Same as add procedure.

The logic flow diagram for the ADD, SUB, ADM, and SBM instructions is shown in Figure 21.

ACL — Add and Carry Logical Word



Description. The C(Y) are added to the C(AC)$_{P,1-35}$. The resultant sum replaces the C(AC)$_{P,1-35}$. The sign

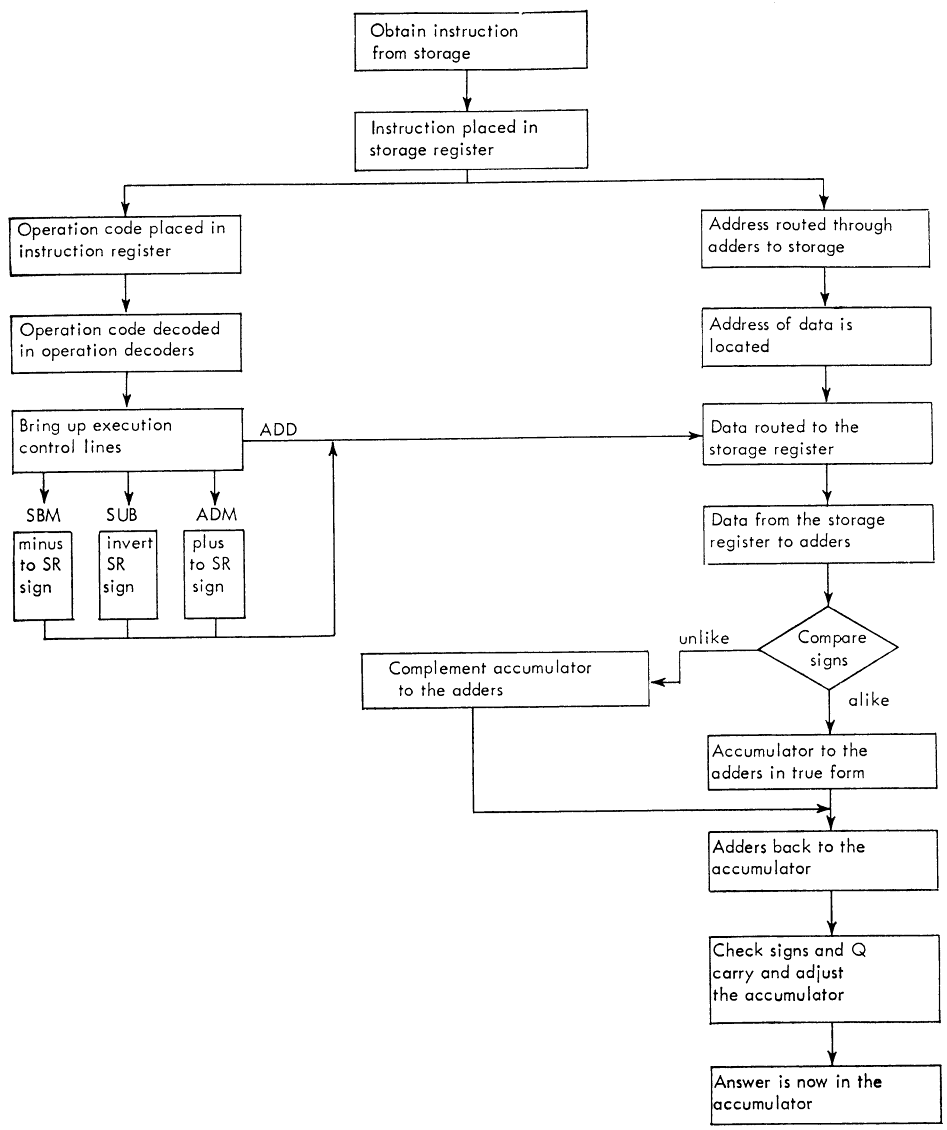

Obtain instruction from storage
Instruction placed in storage register
Operation code placed in instruction register
Operation code decoded in operation decoders
Bring up execution control lines
ADD
SBM
SUB
ADM
minus to SR sign
invert SR sign
plus to SR sign
Address routed through adders to storage
Address of data is located
Data routed to the storage register
Data from the storage register to adders
Compare signs
unlike
alike
Complement accumulator to the adders
Accumulator to the adders in true form
Adders back to the accumulator
Check signs and Q carry and adjust the accumulator
Answer is now in the accumulator

Figure 21. ADD, ADM, SUB, and SBM Flow Chart

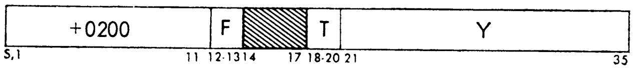
of Y is added to position P of the AC. A carry from AC (P) is added to AC (35). Positions S and Q of the AC are not affected.

Indicators. None.

Timing: 2 cycles

Execution. The C(Y) are taken to the SR. The SR (S, 1-35) are then taken to the adders (P, 1-35). An adder P carry goes to adder 35. Adders (P, 1-35) are then returned to AC (P, 1-35).

MPY — Multiply



Description. The C(Y) are multiplied by the C(MQ). The 35 most significant bits of the 70-bit product replace $C(AC)_{1\text{-}35}$ and the 35 least significant bits replace the $C(MQ)_{1\text{-}35}$. AC (P and Q) are cleared. The signs of the AC and MQ are set to the algebraic sign of the product. The number of bits to the right of the binary point of the first factor added to the number of bits to the right of the binary point of the second factor give the total number of bits to the right of the binary point in the product.

Indicators. None

Timing: 2-14 cycles, modification 1.

Execution. (1) The C(Y) are tested, and if the magnitude of the C(Y) is zero, the C(AC) and C(MQ) are cleared. Step 2 is skipped and step 3 occurs. (2) If the magnitude of the C(Y) is not zero, the $C(AC)_{Q,P,1\text{-}35}$ are cleared and multiplication proceeds:

a. If MQ_{35} contains a 1, the $C(Y)_{1\text{-}35}$ are added to the AC. The $C(AC)_{Q,P,1\text{-}35}$ and the $C(MQ)_{1\text{-}35}$ are then shifted right one position.

b. If MQ_{35} contains a 0, the $C(AC)_{Q,P,1\text{-}35}$ and $C(MQ)_{1\text{-}35}$ are shifted right one position. Step 2 occurs 3 times per cycle on the 7090. With sequential zeros, up to 12 shifts may occur per cycle.

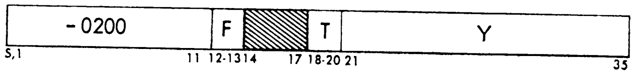
(3) If the signs of the MQ and location Y are the same, the signs of the AC and MQ are made positive. If the signs differ, the signs of the AC and MQ are made negative.

As an example, assume that the AC, MQ, and location Y are four bits in length instead of 35. The following sequence of steps would occur during a multiply. The number 13 is in the MQ and the C(Y) are 6. The actual bit-configuration appears in each register (after the step is complete).

The flow chart is shown in Figure 22.

AC	MQ	Y	COMMENTS
0000	1101	0110	Initial contents of the registers. MQ 35 ready to be tested.
0110	1101		C(Y) added to AC since MQ 35 is a 1.
0011	0110		C(AC, MQ) shifted right one place. Test MQ 35.
0001	1011		No addition, since MQ 35 contained a 0. C(AC, MQ) again shifted right and MQ 35 is tested.
0111	1011		C(Y) added since MQ 35 is a 1.
0011	1101		C(AC, MQ) shifted right and MQ 35 tested.
1001	1101		C(Y) added, since MQ35 is a 1.
0100	1110		C(AC, MQ) shifted right. At this point the shift counter has been reduced to zero and the process stops with the eight-bit product in the AC and MQ registers.

MPR — Multiply and Round



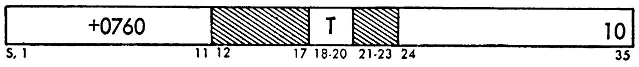
Description. This operation is the same as multiply except that the C(AC) are increased by 1 if MQ (1) contains a one after multiplication is complete.

Indicators. None.

Timing: 2-14 cycles, modification 1.

Execution. (1) Develop the product as in multiply. (2) If MQ (1) contains a 1, add a 1 to AC (35).

RND — Round



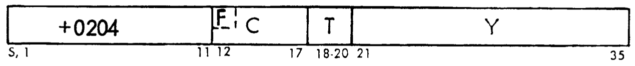
Description. If position 1 of the MQ contains a 1, the C(AC) are increased by one. If MQ (1) contains a 0, the C(AC) are unchanged. In either case the C(MQ) are unchanged. Note that positions 24-35 of this instruction represent part of the operation code. Modification by indexing may change the operation code itself.

Indicators. AC overflow.

Timing: 2 cycles

Execution. If MQ (1) contains a 1, the $C(AC)_{Q\text{-}35}$ is sent to the adders with a carry to adder 35. The adder (Q-35) is then taken to AC (Q-35). If MQ (1) contains a 0, no rounding occurs.

VLM — Variable Length Multiply



Description. This instruction multiplies the C(Y) by the C low-order bits of the C(MQ), to produce a 35 + C

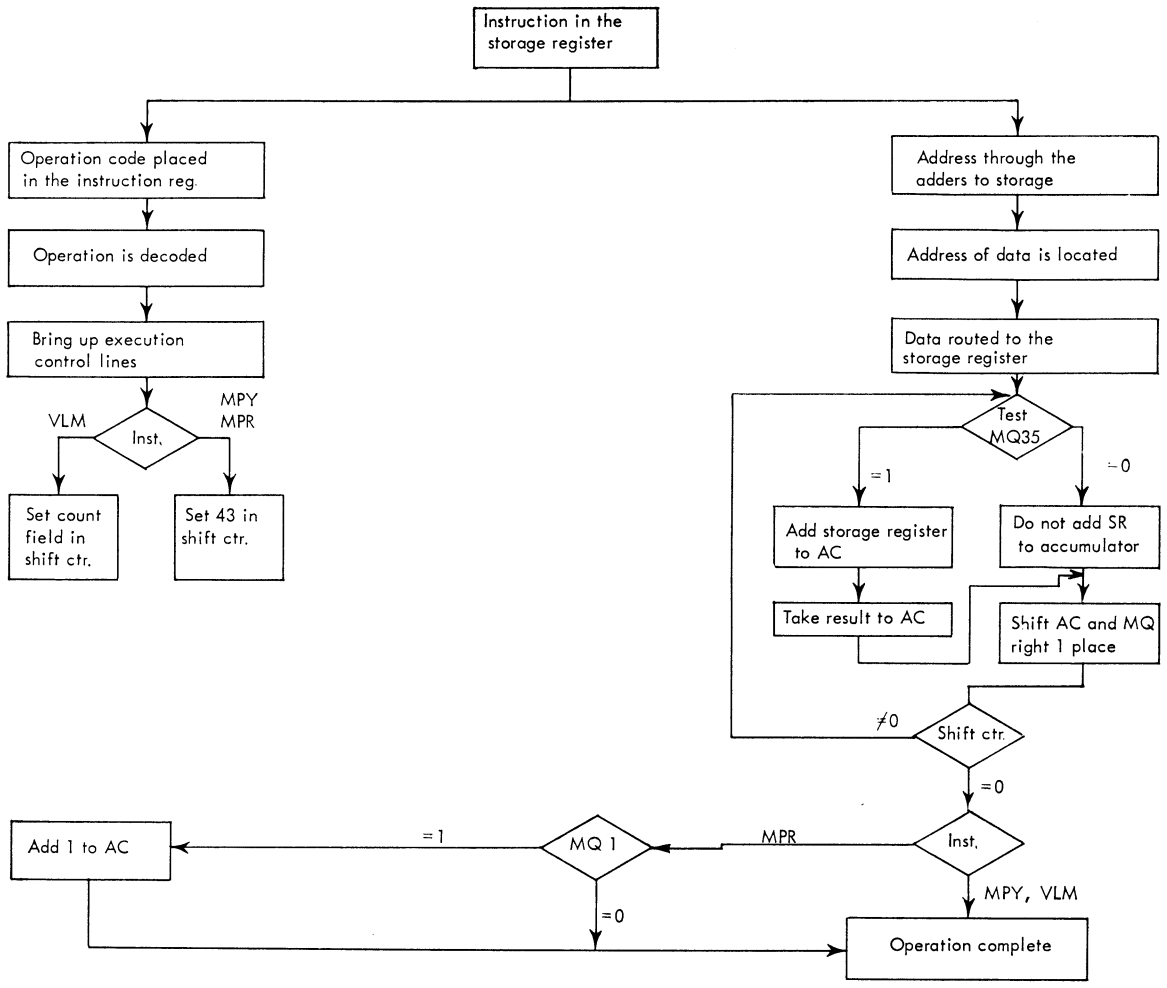

Instruction in the storage register
Operation code placed in the instruction reg.
Operation is decoded
Bring up execution control lines
VLM
Inst.
MPY MPR
Set count field in shift ctr.
Set 43 in shift ctr.
Address through the adders to storage
Address of data is located
Data routed to the storage register
Test MQ35
=1
=0
Add storage register to AC
Do not add SR to accumulator
Take result to AC
Shift AC and MQ right 1 place
≠0
Shift ctr.
=0
Inst.
MPR
MPY, VLM
MQ 1
=1
=0
Add 1 to AC
Operation complete

Figure 22. MPY, MPR, VLM Instruction Flow Chart

bit product. The 35 most significant bits of the product replace the C(AC)$_{1\text{-}35}$ and the C least significant bits replace the C(MQ) 1 through C. Positions Q and P of the AC are cleared. The remaining 35−C positions of the MQ will contain the original 35−C high-order positions of the MQ. The sign of the AC and MQ is the algebraic sign of the product. An example is shown in Figure 23.

If C is zero, the instruction is interpreted as a no-operation and the computer proceeds directly to the next instruction in sequence, leaving the AC unchanged.

If C is not zero but the C(Y) are zero, the C(AC) and C(MQ) are cleared. If the signs of the MQ and location Y are the same, the signs of the AC and MQ are made positive. If the original signs of the SR and MQ differ, the signs of the AC and MQ are made negative. NOTE: A count field which places a 1 bit in both positions 12 and 13 (60 or larger) will cause indirect addressing. In general, counts larger than 35 are meaningless.

Indicators. None.

Timing: 2-14 cycles, modifications 1 and 2

Execution. The instruction is the same as multiply except that the contents of the count field, instead of 43, are placed in the shift counter.

Figure 22 shows the flow chart for MPY, MPR, and VLM instructions.

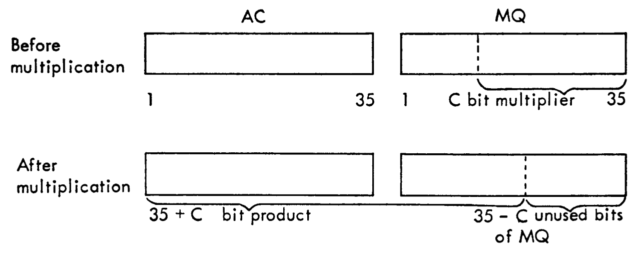
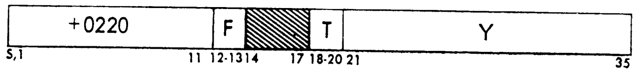


Figure 23. Variable Length Multiply

DVH — Divide or Halt



Description. The c(AC)$_{Q,P,1\text{-}35}$ and the c(MQ)$_{1\text{-}35}$ are treated as a 70-bit dividend plus sign, and the c(Y) as a 35-bit divisor. If the magnitude of c(Y) is greater than the magnitude of c(AC), division takes place. A 35-bit quotient replaces the c(MQ)$_{1\text{-}35}$ and the remainder replaces the c(AC)$_{1\text{-}35}$. The MQ sign is the algebraic sign of the quotient and the AC sign is the sign of the dividend.

If the magnitude of the c (Y) is less than or equal to the magnitude of the c (AC), division does not occur and the computer stops with the divide-check indicator on. For example, if Q or P of the AC contains a 1, the magnitude of the c (Y) is less than the c (AC). If division does not occur, the dividend remains unchanged in the AC and MQ.

Indicators. Divide check

Timing: 3-14 cycles.

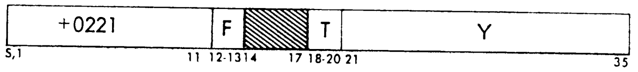
Execution. (1) The c (AC and MQ)$_{1\text{-}35}$ are shifted left one position, creating a zero in position 35 of the MQ. (2) If the magnitude of the c (Y) is less than or equal to the magnitude of c (AC), the magnitude of c (Y) is subtracted from the magnitude of c (AC) and a one replaces the zero in MQ$_{35}$. Step 1 is then repeated (Figure 24). (3) If the magnitude of the c (Y) is greater than the magnitude of the c(AC), the computer returns to step 1.

The above process occurs 35 times for each division, three times per machine cycle.

The following example is a division problem. Again assume a four-bit machine. The problem is 66 divided by 5, and the binary numbers represent the result of the described step.

AC	MQ	Y	COMMENTS
0100	0010	0101	Initial contents. c (AC) less than c (Y); division will take place.
1000	0100		c (AC and MQ) shifted left one place; c (AC) greater than c (Y).
0011	0101		c (Y) subtracted from c (AC) and a 1 replaces MQ 35.
0110	1010		c (AC and MQ) shifted left one place; c (AC) greater than c (Y).
0001	1011		c (Y) subtracted from c (AC) and a 1 replaces MQ 35.
0011	0110		c (AC and MQ) shifted left one place; c (AC) less than c (Y).
0110	1100		c (AC and MQ) shifted left one place; c (AC) greater than c (Y).
0001	1101		c (Y) subtracted from c (AC) and a 1 replaces MQ 35.
			The quotient is now complete in the MQ with the remainder in the AC.

DVP — Divide or Proceed



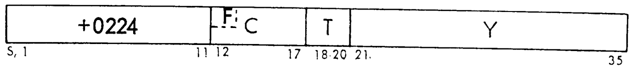
Description. If the magnitude of the c (Y) is greater than the magnitude of the c (AC), division occurs as with the DVH instruction. If the magnitude of the c (Y) is less than or equal to the magnitude of the c (AC), the divide-check indicator is turned on and the computer proceeds to the next instruction.

Indicators. Divide check

Timing: 3-14 cycles.

Execution. Exactly the same as DVH except that instead of halting, when a divide-check occurs, the computer executes the next sequential instruction (Figure 24).

VDH — Variable Length Divide or Halt



Description. This instruction is the same as a DVH except that a C-bit quotient plus sign replaces the C low-order positions of the MQ. The remainder replaces the c (AC)$_{1\text{-}35}$ and the 35—C high-order positions of the MQ. Instead of 43 being placed in the shift counter initially, C is placed there. If C is zero the instruction is interpreted as a no-operation and the computer proceeds directly to the next instruction in sequence.

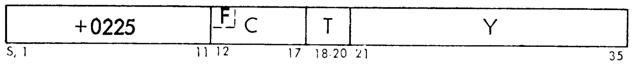
Indicators. Divide check.

Timing: 2-14 cycles, modification 2

Execution. The same operation as DVH except as noted above.

NOTE: Indirect addressing may occur if the count field places 1-bits in positions 12 and 13 of the instruction (Figure 24).

VDP — Variable Length Divide or Proceed



Description. This instruction is the same as DVP except that a C bit quotient with a sign replaces the C low-order positions of the MQ. The remainder replaces the c (AC)$_{1\text{-}35}$ and the 35–C high-order positions of the MQ. C rather than 43 is placed in the shift counter initially. If C is zero the instruction is interpreted as a no-operation and the computer proceeds directly to the next instruction in sequence.

Indicators. Divide check.

Timing: 2-14 cycles, modification 2

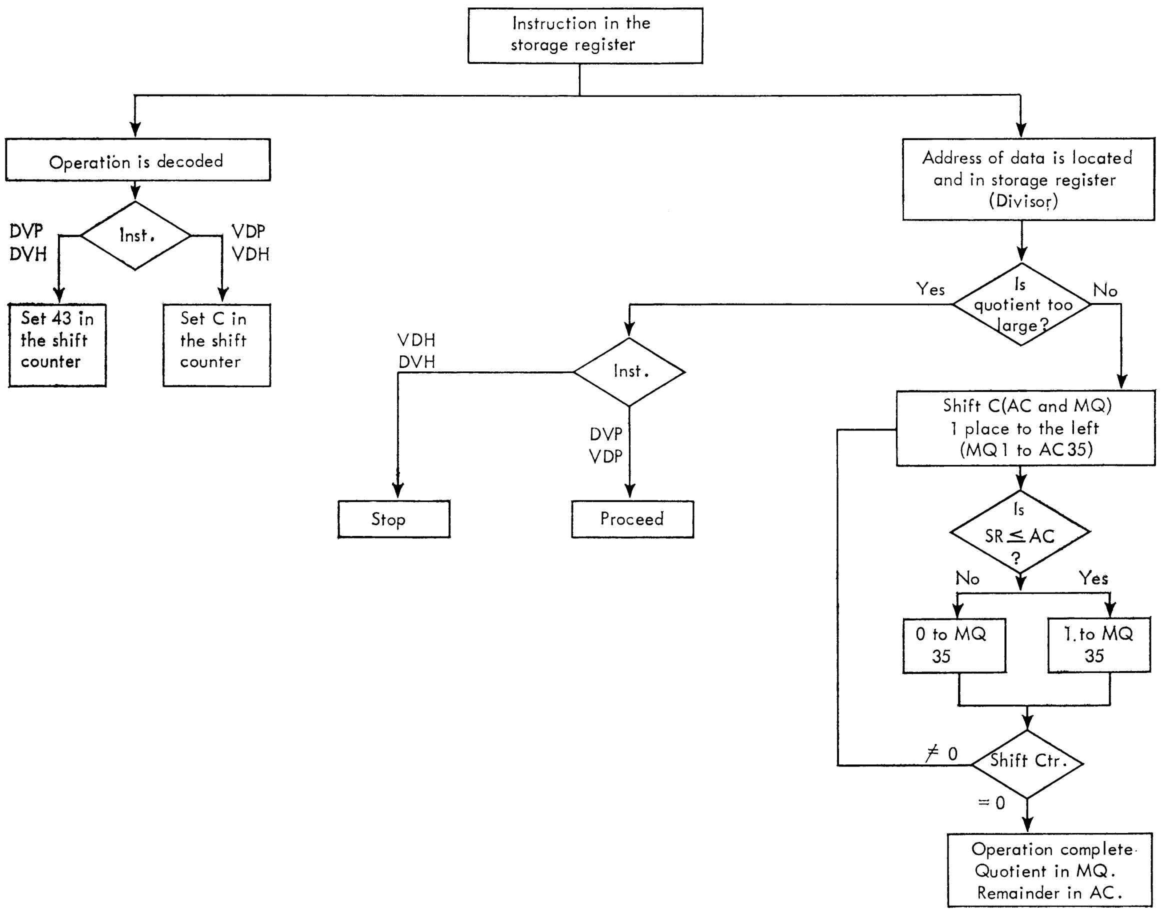

Instruction in the storage register
Operation is decoded
Inst.
DVP DVH
VDP VDH
Set 43 in the shift counter
Set C in the shift counter
Address of data is located and in storage register (Divisor)
Is quotient too large?
Yes
No
VDH DVH
Inst.
DVP VDP
Stop
Proceed
Shift C(AC and MQ) 1 place to the left (MQ1 to AC35)
Is SR ≤ AC ?
No
Yes
0 to MQ 35
1 to MQ 35
≠ 0
Shift Ctr.
= 0
Operation complete. Quotient in MQ. Remainder in AC.

Figure 24. DVH, DVP, VDH, and VDP Flow Chart

Execution. The same procedure as DVP except as noted above.

NOTE: Indirect addressing may occur if the count field places 1 bits in positions 12 and 13 of the instruction.

Figure 24 shows the logic flow chart for DVH, DVP, VDH, and VDP instructions.

Floating Point Operations

The following operations are divided into two groups to describe the processing of floating-point numbers in either normalized or unnormalized form. The possibility of floating-point overflow or underflow during the execution of a floating-point instruction is indicated by an asterisk (*). All conditions of underflow and overflow are discussed following the last floating-point instruction.

Floating Point Arithmetic

The algebraic addition of two floating-point numbers in the computer is analogous to the ordinary algebraic addition of two signed numbers with decimal points. An example is the algebraic addition of the two numbers 100 and −0.1009:

$$\begin{array}{r} 100.0000 \\ -\ \ 0.1009 \\ \hline 99.8991 \end{array}$$

Note that the second number must be shifted to the right to line up the decimal points, and that the first number must be supplied with additional zeros. The same addition performed with numbers expressed in floating-point decimal form, would be:

$$\begin{array}{r} .1000 \times 10^{3} \\ -.1009 \times 10^{0} \end{array}$$

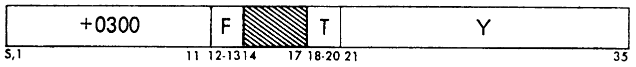
Again, before the addition, the lower number is shifted to the right with a compensating change in the exponent and corresponding zeros are added to the number on the upper line:

$$\begin{array}{r} .1000000 \times 10^3 \\ -.0001009 \times 10^3 \\ \hline .0998991 \times 10^3 = .998991 \times 10^2 \end{array}$$

Note also that the digits of the answer must be moved to the left to be in normalized form and that the final fraction contains more digits than either of the two numbers involved in the addition.

In the computer the two numbers are expressed as binary fractions, each having an 8-bit binary characteristic to represent the exponent of 2. The "lining up" is done by shifting from the AC into the MQ. The result of an addition or multiplication is normalized by shifting the fractions in the AC and MQ left while making compensating changes in the characteristic of the sum or product.

FAD — Floating Add



Description. The floating-point numbers located in Y and the AC are added together. The most significant portion of the result appears as a normal floating-point number in the AC. The least significant portion of the result appears in the MQ as a floating-point number with a characteristic 33 (octal) less than the AC characteristic. The signs of the AC and MQ are set to the sign of the larger factor. The sum in the AC and MQ is always normalized whether the original factors were normal or not. If C(AC)$_{1-35}$ contain zeros, the FAD may be used to normalize an unnormal floating-point number.

Indicators. Floating-point underflow, overflow, and floating-point trap.

Timing: 6-15 cycles, modification 3

Execution

1. The MQ register is cleared to zeros.

2. The C(Y) are placed in the SR.

3. If the characteristic in the SR is less than the characteristic in the AC, the C(SR) and C(AC)$_{S,1-35}$ are interchanged, as the number with the smaller characteristic must appear in the AC before addition can take place.

4. The MQ is given the same sign as the AC.

5. If the difference in the characteristics is greater than 63, the C(AC) are cleared. If the difference in the characteristics is a number N less than or equal to 63, the C(AC)$_{9-35}$ are shifted right N places. Bits shifted out of position 35 of the AC enter position 9 of the MQ. Bits shifted out of position 35 of the MQ are lost.

6. The characteristic in the SR replaces the C(AC)$_{1-8}$.

7. The C(SR)$_{9-35}$ are added to the C(AC)$_{9-35}$ and this sum replaces the C(AC)$_{9-35}$. If the signs of the AC and SR are unlike, the C(SR)$_{9-35}$ are added to the 1's complement of the C(AC)$_{9-35}$. Since the C(AC)$_{9-35}$ represent a pure fraction, the magnitude of their 1's complement is equal to $(1 - 2^{-27}) -$ C(AC)$_{9-35}$.

8. Regardless of the sign or relative magnitudes of the SR and AC, the result appears in double-precision form with signs alike in both the AC and MQ. If the signs of the AC and SR are the same and the magnitude of the sums of the fractions is greater than or equal to one, there is a carry from position 9 into position 8 of the AC. Thus, the characteristic of the AC is increased by one.* In this event, the fractions of the AC and MQ are shifted right one position and a 1 is inserted into position 9 of the AC. If the signs of the AC and SR are different, there are two cases, both depending on the difference between the SR and AC fractions.

Case 1. If the magnitude of the SR fraction is greater than the fraction in the AC, the AC and MQ signs are both changed to the sign of the SR. If the fraction of the MQ is zero, the difference between the fractions of the SR and AC is placed in the AC. If the fraction of the MQ is not zero, the difference between the fractions of the SR and AC, minus one, is placed in the AC; the 2's complement of the MQ fraction replaces the fraction in the MQ.

Case 2. If the magnitude of the SR fraction is less than the fraction in the AC, the difference of the two fractions replaces the fraction of the AC. The sign of the AC and the entire MQ remain unchanged.

9a. If the resulting fractions in both the AC and MQ are zero, the AC is cleared, yielding a normal zero. If the fractions are in normalized form before the FAD is given, this result can only occur if the signs are different and the C(Y)$_{1-35}$ are equal to the C(AC)$_{1-35}$. The signs of the AC and MQ will be equal to the sign of the number originally in the AC. If the resulting fraction in the AC is zero and the two numbers were not in normalized form before addition, the signs of the AC and MQ are equal to the sign of the original number having the smaller characteristic.

9b. If the resulting fractions in the AC and MQ are not zero, the fractions of the AC and MQ are shifted left until a 1 appears in position 9 of the AC. Bits enter position 35 of the AC from position 9 of the MQ. The characteristic in the AC is reduced by one for each position shifted.* No shifting is necessary if the fraction of the AC is in normal form at the beginning of this step.

10. The MQ is given a characteristic which is 27 less than the characteristic in the AC,* unless the AC contains a normal zero, in which case zeros are left in positions 1-8 of the MQ.

If the P and/or Q positions of the AC are not zero before the execution of the FAD, the result will usually be incorrect. Non-zero bits in P and/or Q which are initially interpreted as part of the AC characteristic make it larger than the characteristic in the SR so that the interchange in step 3 will always take place. During the interchange a 1 will be placed in position S of the SR if there is a 1 in either S or P positions of the AC, so that the sign of the number may be changed. Any bit in Q is lost during the interchange and both P and Q are cleared when the C (SR) replace the C (AC). The difference between the two characteristics is computed after the interchange occurs, so that in step 5, N will not be equal to the difference between the original characteristics. In step 6 the characteristic in the SR, with its Q and P bits missing, replaces the characteristic in the AC. Consider as a sample problem the addition of:

$$\begin{array}{ll} 2^2 \times .1001 = & \text{(SR)} +10000010.1001 \\ 2^5 \times .1001 = & \text{(AC)} +10000101.1001 \end{array}$$

First, the exponents must be equalized and then the addition may proceed. The characteristics are checked and found unequal, with the largest in the AC. The numbers in the AC and SR are then exchanged, giving:

SR	+10000101.1001
AC	+10000010.1001

The MQ content is zeros at this time. The $\text{C (AC)}_{9\text{-}35}$ are then shifted right the number of places needed to equalize the exponents. (Remember that the binary point is located between positions 8 and 9 of all registers.) The registers then appear as:

SR	+10000101.1001
AC	+10000101.0001
MQ	+00000000.0010

The fractions (positions 9-35) may now be added.

SR	+10000101.1001
AC	+10000101.1010
MQ	+00000000.0010

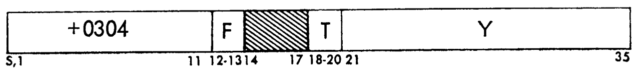
AC position 9 is checked for a 1 and no normalizing occurs. The MQ characteristic is now set. It is equal to the AC characteristic minus the number of places in the AC fraction (27 in the computer, 4 in this example):

SR	+10000101.1001
AC	+10000101.1010
MQ	+10000001.0010

Decoding the results into the original format, we find:

$$\begin{array}{l|l} 2^5 \times .1001 & \text{MQ} = 2^1 \times .0010 = 2^5 \times .00000010 \\ \underline{2^5 \times .0001001} & \underline{\text{AC} = \qquad\qquad 2^5 \times .1010} \\ 2^5 \times .1010001 & \text{resultant sum} = 2^5 \times .10100010 \end{array}$$

FAM — Floating Add Magnitude



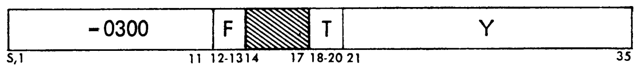
Description. This instruction algebraically adds the positive magnitude of the floating-point numbers contained in Y to the signed floating-point number in the AC. The sum is normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 6-15 cycles, modification 3

Execution. The same procedure as FAD except that the magnitude of the number in the SR is used (SR sign is forced plus).

UFA — Unnormalized Floating Add



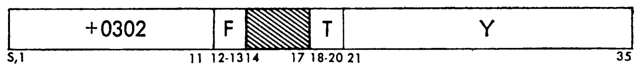
Description. This instruction algebraically adds two floating-point numbers contained in the AC and Y. The sum is not normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 5-10 cycles, modification 4

Execution. The same procedure as FAD except that no normalizing will occur (step 9).

FSB — Floating Subtract



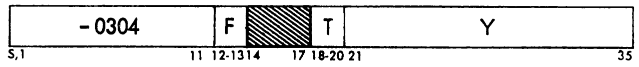
Description. This instruction algebraically subtracts the floating-point number located in Y from the floating-point number in the AC, and normalizes the result.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 6-15 cycles, modification 3

Execution. The same procedure as FAD except that the negative of the C(Y) are placed in the SR (SR sign is reversed).

UAM — Unnormalized Add Magnitude



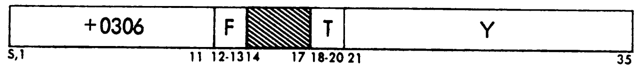
Description. This instruction algebraically adds the magnitude of the floating-point number contained in Y to the signed floating-point number in the AC. The sum is not normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 5-11 cycles, modification 4

Execution. The same procedure as FAD except that the sign of the number in the SR is made positive and the result is not normalized.

FSM — Floating Subtract Magnitude



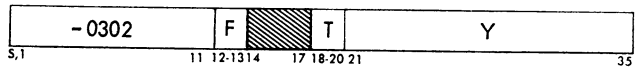
Description. This instruction algebraically subtracts the magnitude of a floating-point number stored at Y from the signed floating-point number in the AC. The result is normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 6-15 cycles, modification 3

Execution. The same procedure as FAD except that the negative magnitude of the contents of Y are used (SR sign is forced minus).

UFS — Unnormalized Floating Subtract



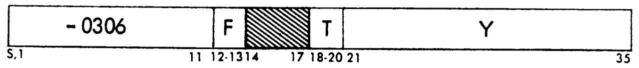
Description. This instruction algebraically subtracts the floating-point number located in Y from the floating-point number in the AC. The result is not normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 5-10 cycles, modification 4

Execution. The same procedure as FAD except that the negative of the contents of Y are placed in the SR and normalizing does not occur.

USM — Unnormalized Subtract Magnitude



Description. This instruction algebraically subtracts the magnitude of a floating-point number stored at Y from the signed floating-point number in the AC. The result is not normalized.

Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

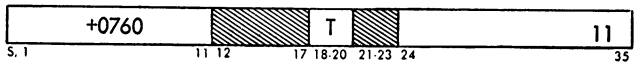
Timing: 5-11 cycles, modification 4

Execution. The same procedure as FAD except that the negative magnitude of the contents of Y are used and the result is not normalized.

The differences between answers, received after execution of a floating-point add or subtract operation, when using a 704 or 7090 system is a matter of increased precision and is summarized as:

	704	7090
1.	Accumulator sign and MQ sign not necessarily the same.	Accumulator sign and MQ sign are guaranteed to be equal.
2.	Characteristic difference between accumulator and MQ is usually 27_{10}, but it can be 28_{10} when adding numbers of unlike signs.	Characteristic difference between accumulator and MQ is always 27_{10}.
3.	If the accumulator is zero and the MQ is not, the sum will not be shifted and the accumulator will be made equal to a normal zero.	If the accumulator is zero and the MQ is not, the MQ factor will be shifted in order to normalize the sum.

FRN — Floating Round



Description. Floating-point add, subtract, and multiply produce a double-word result. The instruction

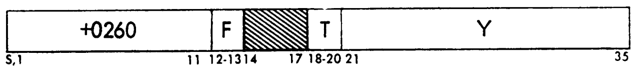
FRN will add 1 to position 35 of the AC if the MQ fraction is equal to or exceeds half the magnitude of a 1-bit in AC 35. The AC is corrected if rounding results in a carry from AC 9.

Indicators. Floating-point overflow, floating-point trap.

Timing: 2 cycles

Execution. If MQ 9 contains a 1, a carry will be added to AC 35. A carry out of AC 9 increases the characteristic of the AC by 1, causes the fraction of the AC to be shifted right and a 1 to be placed in AC 9. Since the address part of this instruction represents part of the operation code, any modification by an index register may result in changing the operation itself.

FMP — Floating Multiply



Description. The C(Y) are multiplied by the C(MQ). The most significant part of the product appears in the AC and the least significant part appears in the MQ. The product of two normalized numbers is in normalized form. If either of the numbers is not normalized, the product may or may not be in normalized form.

Indicators. Floating-point underflow, floating-point overflow, and floating-point trap.

Timing: 2-13 cycles, modification 1

Execution

1. The C(Y) are placed in the SR and the AC is cleared.

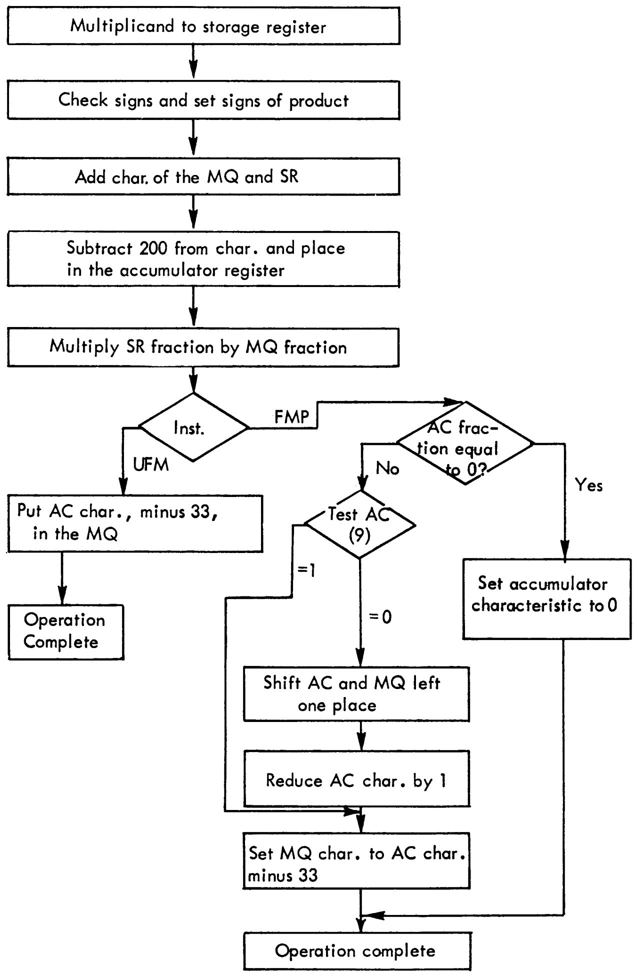
2a. If the multiplicand is a normal zero so that C(SR)$_{1\text{-}35}$ are equal to zero, the MQ$_{1\text{-}35}$ is cleared and the calculator proceeds directly to the next instruction in sequence. This is also true on the 7090 if the MQ fraction is zero.

2b. If the C(SR)$_{9\text{-}35}$ are not equal to zero, the sum of the characteristics in the SR and MQ minus 128 is placed in positions 1-8 of the AC* (Figure 25).

3. The C(SR)$_{9\text{-}35}$ are multiplied by the C(MQ)$_{9\text{-}35}$. The 27 most significant digits of the 54-digit product replace the C(AC)$_{9\text{-}35}$ and the 27 least significant digits replace the C(MQ)$_{9\text{-}35}$. The sign of the AC is the algebraic sign of the product.

4a. If the fraction in the AC is zero, the C(AC)$_{Q,\ P,\ 1\text{-}35}$ are cleared, yielding a signed normal zero.

4b. If the position 9 of the AC contains a zero but the fraction in the AC is not zero, the C(AC)$_{10\text{-}35}$ and



Multiplicand to storage register
Check signs and set signs of product
Add char. of the MQ and SR
Subtract 200 from char. and place in the accumulator register
Multiply SR fraction by MQ fraction
Inst.
FMP
UFM
AC fraction equal to 0?
No
Yes
Put AC char., minus 33, in the MQ
Test AC (9)
=1
=0
Set accumulator characteristic to 0
Operation Complete
Shift AC and MQ left one place
Reduce AC char. by 1
Set MQ char. to AC char. minus 33
Operation complete

Figure 25. FMP and UFM Flow Chart

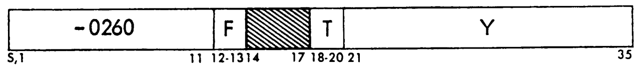
the C(MQ)$_{9\text{-}35}$ are shifted left one position and the characteristic in the AC is reduced by 1.

5a. If the AC contains a normal zero, positions 1-8 of the MQ are cleared.

5b. If the AC does not contain a normal zero, the C(MQ)$_{1\text{-}8}$ are replaced by a characteristic which is 27 less than the characteristic in the AC (*).

6. The sign of the MQ is replaced by the sign of the AC.

UFM — Unnormalized Floating Multiply



Description. This instruction multiplies the floating-point number at Y by the floating-point number in the MQ. The result is not normalized.

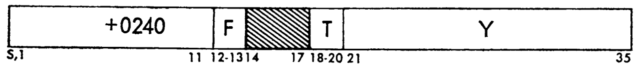
Indicators. Floating-point underflow and floating-point overflow; floating-point trap.

Timing: 2-13 cycles, modification 1

Execution. The same procedure as FMP except that the product is not normalized or zero tested.

Figure 25 shows a flow chart of the FMP and UFM instructions.

FDH — Floating Divide or Halt



Description. The C(AC) are divided by the C(Y). The quotient appears in the MQ and the remainder appears in the AC. If the magnitude of the AC fraction is greater than or equal to twice that of the C(Y)$_{9-35}$, or if the magnitude of the C(Y)$_{9-35}$ is zero, the divide check indicator is turned on and the computer stops, leaving the dividend in the AC unchanged and a normal zero in the C(MQ). The quotient is in normal form if both the dividend and divisor are in that form. If they are, the magnitude of the ratio of the fraction in the AC to the fractional part of C(Y) is less than two but greater than one-half.

Indicators. Floating-point underflow, floating-point overflow, divide check, and floating-point trap.

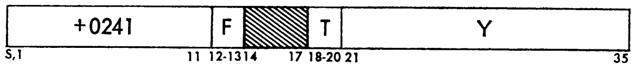
Timing: 3-13 cycles, modification 5

Execution

1. The C(Y) are placed in the storage register.
2. The MQ is cleared.
3. The sign of the MQ is made equal to the algebraic sign of the quotient. The sign of the AC remains unchanged throughout so that the signs of the remainder and dividend always agree.
4. If the magnitude of the fraction in the AC is greater than or equal to twice the magnitude of the fraction in the SR, or if the fraction in the SR is zero, the divide-check indicator and panel light are turned on, the calculator stops and the dividend is left unchanged in the AC.
5. If the fraction in the AC is zero, the C(AC)$_{Q,P,1-35}$ are cleared and the remaining steps are skipped. If S(AC) is minus, the sign is forced plus.
6. If the magnitude of the fraction in the AC is greater than or equal to the magnitude of the fraction in the SR, the AC is shifted right one position, and the characteristic in the AC is increased by one.* The bit in position 35 of the AC enters position 9 of the MQ.
7. The characteristic of the AC minus the characteristic of the SR plus 128 is placed in positions 1-8 of the MQ.*
8. The fractional part of the dividend, which consists of the C(AC)$_{9-35}$ (and the C(MQ) if the condition of step 6 is met), is divided by the fraction in the SR and the quotient replaces the C(MQ)$_{9-35}$.
9. The 27-bit remainder resulting from the division in step 8 replaces the C(AC)$_{9-35}$.
10. The characteristic in the AC is reduced by 27*.

NOTE: Even though the numbers are not in normalized form, the quotient will be normalized if the ratio above holds. If the fraction in the AC is zero, a normalized zero will result in the MQ.

FDP — Floating Divide or Proceed



Description. This instruction divides the floating-point number stored in the AC by the floating-point number located at Y.

Indicators. Floating-point underflow, floating-point overflow, divide check, and floating-point trap.

Timing: 3-13 cycles, modification 5

Execution. The same procedure as FDH except that if the computer cannot handle the problem, it does not halt but proceeds to the next sequential instruction.

Floating-Point Trap

During the execution of floating-point instructions the resultant characteristic in the AC and MQ may exceed eight bit positions (result is too large for storage). The capacity is exceeded if the exponent goes beyond +177 or below −200. Beyond +177 is termed *overflow* while below −200 is termed *underflow.* Overflow and underflow may occur in either the AC or the MQ registers.

To aid the programmer in checking for these conditions, a unique check called *floating-point trap* is used. The computer will, upon sensing an underflow or overflow, put the address plus one of the instruction that caused the condition into the address portion of location 0000.

An identifying code, telling whether an underflow or an overflow occurred and whether the most significant result is in the AC or MQ, is placed in the decrement portion of location 0000. The computer then executes the instruction at location 0010 and proceeds from there. These underflows and overflows are termed *spills.* The decrement positions and meaning of a 1-bit in these positions is:

BIT POS.	MEANING
14.	Divide only (MQ register is not an extension of the AC factor).
15.	Overflow in either ACC or MQ, (or both) registers.
16.	AC factor exceeded.
17.	MQ fraction is excessive.

Shifting Operations

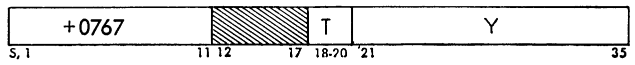
Shift instructions are used to move the contents of the AC and/or the MQ either to the right or the left of their original positions. With the exception of the ROTATE MQ LEFT instruction, zeros are automatically introduced in the vacated positions of a register. Thus, a shift larger than the bit capacity of the register will cause the contents of the register to be replaced by zeros.

When a shift instruction is interpreted, the amount of the shift is determined by bit positions 28-35 of the instruction. This provides a maximum shift of 377 places. Any number larger than 377 is interpreted as modulo 400. By modulo 400 is meant that, given any shift count, the actual number of positions shifted will be the remainder after dividing the shift count by 400.

All shift instructions are subject to address modification through indexing. Shifting a number in a register is equivalent to multiplying or dividing it by a power of 2 (as long as none of the significant bits is lost).

In the following description of the shift instructions, the number of positions to be shifted is specified by "positions 28-35." With indexing, this shift is modified by positions 10-17 of the specified index register or registers.

ALS — Accumulator Left Shift



Description. This instruction causes the $C(AC)_{Q,\,P,\,1\text{-}35}$ to be shifted left the number of places specified in positions 28-35 of the address portion of the instruction. The sign position is unchanged.

Indicators. AC overflow.

Timing: 2-4 cycles, modification 7

Execution. If a non-zero bit is shifted into position P from position 1, the AC overflow indicator is turned on. Bits shifted past position Q are lost. Vacated positions are filled with zeros (Figure 26).

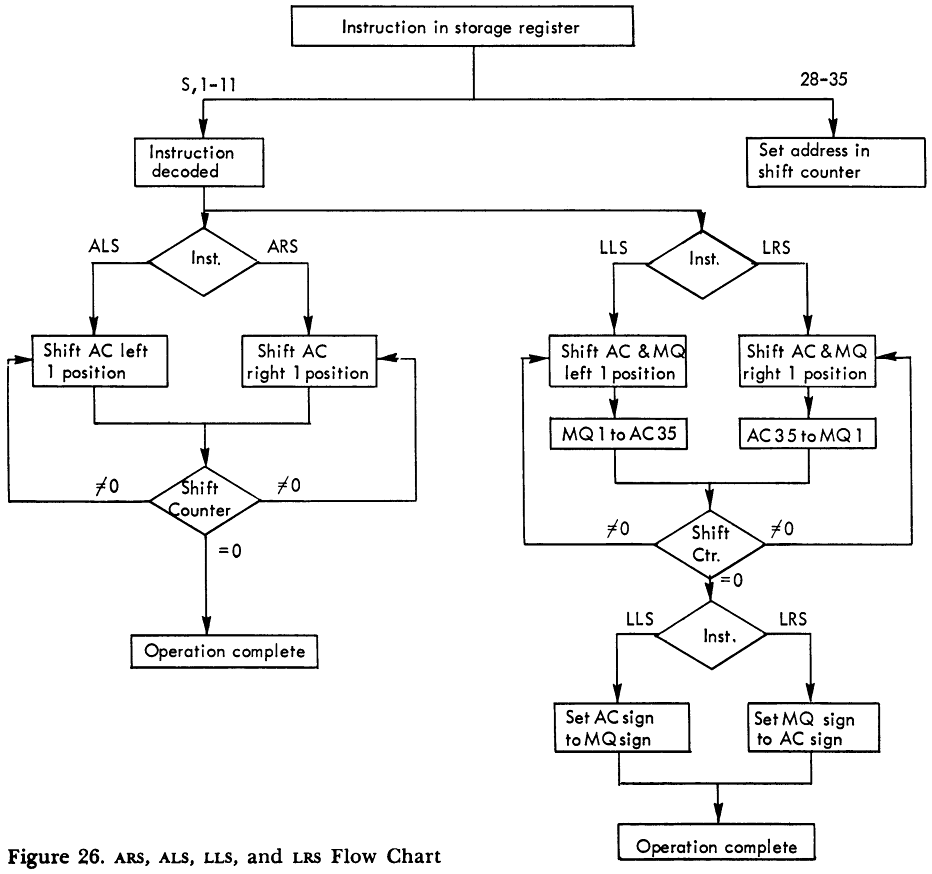


Figure 26. ARS, ALS, LLS, and LRS Flow Chart

ARS — Accumulator Right Shift

+0771	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (AC)$_{Q, P, 1\text{-}35}$ are shifted right the number of places specified in positions 28-35 of the address portion of the instruction. The sign position is unchanged.

Indicators. None.

Timing: 2-4 cycles, modification 7

Execution. Bits shifted past position 35 of the accumulator are lost. Bits shifted from Q enter P and bits from P enter position 1. Vacated positions are filled with zeros (Figure 26).

LLS — Long Left Shift

+0763	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (AC)$_{Q, P, 1\text{-}35}$ and the C (MQ)$_{1\text{-}35}$ are treated as one register. The contents of these registers are shifted left the number of places specified in positions 28-35 of the address portion of the instruction. The MQ sign position is unchanged and the sign of the AC is made to agree with it.

Indicators. AC overflow.

Timing: 2-7 cycles, modification 7

Execution. Bits enter position 35 of the AC from position 1 of the MQ. If a non-zero bit is shifted into or through position P, the AC overflow indicator is turned on. Bits shifted past position Q are lost. Positions vacated are filled with zeros (Figure 26).

LRS — Long Right Shift

+0765	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (AC)$_{Q, P, 1\text{-}35}$ and the C (MQ)$_{1\text{-}35}$ are treated as one register. The contents of these registers are shifted right the number of places specified in positions 28-35 of the address portion of the instruction. The AC sign is unchanged and the sign of the MQ is made to agree with it.

Indicators. None.

Timing: 2-7 cycles, modification 7

Execution. Bits enter position 1 of the MQ from position 35 of the AC. Bits shifted past position 35 of the MQ are lost. Vacated positions are filled with zeros (Figure 26).

Figure 26 shows the flow chart for the ARS, ALS, LLS, and LRS instructions.

LGL — Logical Left Shift

-0763	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (AC)$_{Q, P, 1\text{-}35}$ and the C (MQ)$_{S, 1\text{-}35}$ are treated as one register. Their contents are shifted left the number of places specified in positions 28-35 of the address portion of the instruction. The sign of the AC is unchanged.

Indicators. AC overflow.

Timing: 2-7 cycles, modification 7

Execution. Bits enter position S of the MQ from position 1 of the MQ. Bits from MQ (S) then enter position 35 of the accumulator. If a non-zero bit is shifted into or through position P of the AC, the AC overflow indicator is turned on. Bits are shifted from P to Q and any bits shifted from Q are lost. Vacated positions are filled with zeros.

LGR — Logical Right Shift

-0765	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (AC)$_{Q, P, 1\text{-}35}$ and the C (MQ)$_{S, 1\text{-}35}$ are treated as one register. Their contents are shifted right the number of places specified in positions 28-35 of the address portion of the instruction. The sign of the AC is unchanged.

Indicators. None.

Timing: 2-7 cycles, modification 7

Execution. Bits enter position S of the MQ from position 35 of the AC. Bits enter MQ 1 from MQ (S). Bits shifted past position 35 of the MQ are lost. Vacated positions are filled with zeros.

RQL — Rotate MQ Left

-0773	▨	T	Y
S, 1 — 11	12 — 17	18-20	21 — 35

Description. The C (MQ) are shifted left the number of places specified by positions 28-35 of the address portion of the instruction. The instruction shifts position S into position 35, and thus the register becomes a circular one.

Indicators. None.

Timing: 2-4 cycles, modification 7

Execution. Bits are rotated from position 1 of the MQ to position S, and from position S to position 35. No bits are lost.

Word Transmission Operations

The operations described in this section are concerned with the movement of words or parts of words from one core location or register to another.

LDQ — Load MQ

+0560	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. This instruction places the contents of Y into the MQ. The C (Y) are unchanged.

Indicators. None.

Timing: 2 cycles

STQ — Store MQ

-0600	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. This instruction places the contents of the MQ into the specified Y location. The C (MQ) remain unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

SLQ — Store Left Half MQ

-0620	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. The C (MQ) $_{S, 1\text{-}17}$ replace the C (Y) $_{S, 1\text{-}17}$. The C (MQ) and the C (Y) $_{18\text{-}35}$ are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STO — Store

+0601	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. The C (AC) $_{S, 1\text{-}35}$ replace the C (Y). The C (AC) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

SLW — Store Logical Word

+0602	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. The C(AC)$_{P,1\text{-}35}$ replace the C(Y). The C (AC) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STP — Store Prefix

+0630	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. The C (AC) $_{P, 1, 2}$ replace the C (Y) $_{S, 1, 2}$. The C (Y) $_{3\text{-}35}$ and the C (AC) are unchanged.

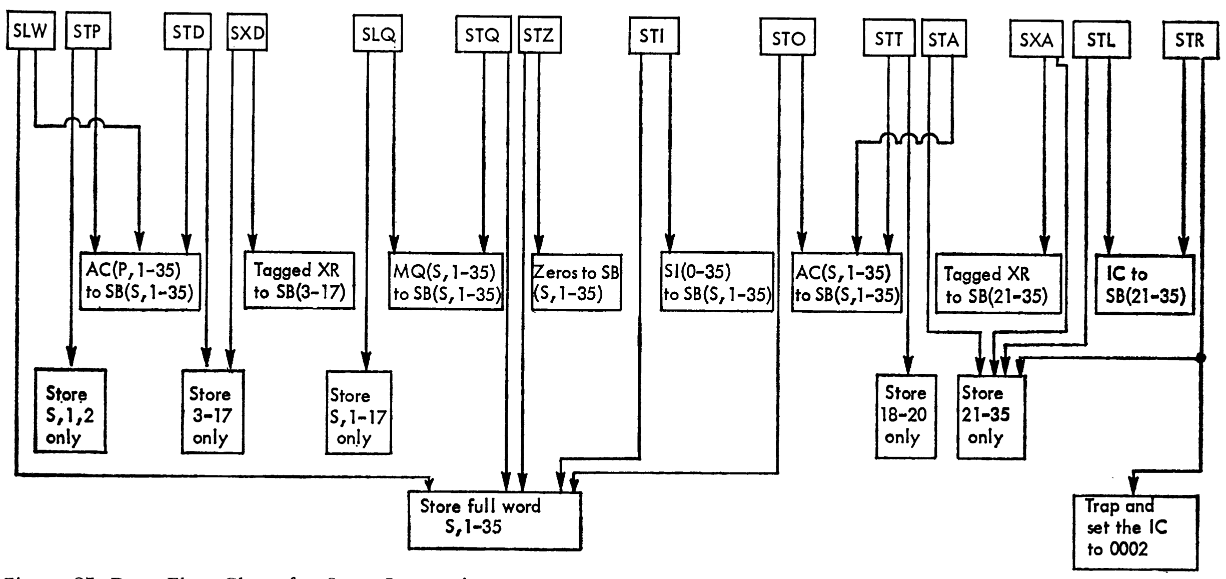


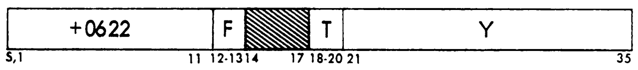
Figure 27. Data Flow Chart for Store Instructions

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STD — Store Decrement



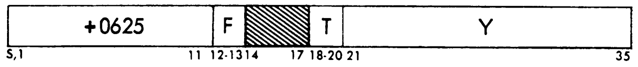
Description. The C (AC) $_{3-17}$ replace the C (Y) $_{3-17}$. The C (Y) $_{S, 1, 2, 18-35}$ and the C (AC) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STT — Store Tag



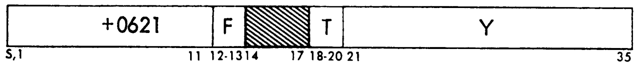
Description. The C (AC) $_{18-20}$ replace the C (Y) $_{18-20}$. The C (Y) $_{S, 1-17, 21-35}$ and the C (AC) remain unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STA — Store Address



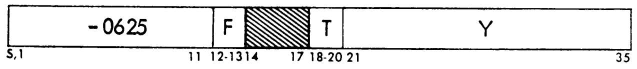
Description. The C (AC) $_{21-35}$ replace the C (Y) $_{21-35}$. The C (Y) $_{S, 1-20}$ and the C (AC) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

STL — Store Instruction Location Counter



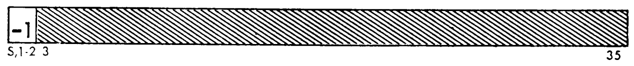
Description. The location of the STL instruction plus 1 replaces the C (Y) $_{21-35}$. The C (Y) $_{S, 1-20}$ are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. Instruction counter contents to address switches. Address switches to the storage register bus. Storage register (21-35) to storage. See Figure 27.

STR — Store Location and Trap



Description. The location of the STR instruction, plus one, replaces positions 21-35 of location 0000. The computer then takes its next instruction from location 0002. The contents of positions 3-35 of this instruction are not interpreted by the computer.

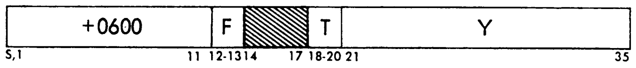
Indicators. None.

Timing: 2 cycles

Execution.

NOTE: Conflicts may arise when the computer is operated in the trapping mode. This instruction, transfers in the trap mode, and floating-point trap, all use location 0000. See Figure 27.

STZ — Store Zero



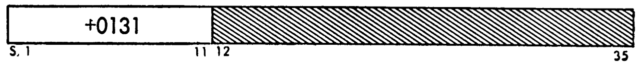
Description. The C (Y) $_{1-35}$ are replaced by zeros and the C (Y) $_{S}$ are made plus.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 27.

XCA — Exchange AC and MQ



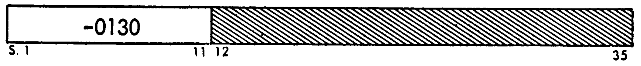
Description. The C (AC) $_{S, 1-35}$ are exchanged with the C(MQ)$_{S,1-35}$. Positions P and Q of the AC are cleared.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 28.

XCL — Exchange Logical AC and MQ



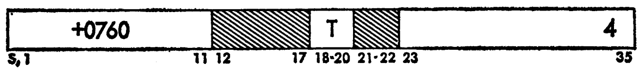
Description. The C (AC) $_{P, 1-35}$ are exchanged with the C (MQ) $_{S, 1-35}$. Positions S and Q of the AC are cleared.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 28.

ENK — Enter Keys



Description. This instruction places the contents of 36 panel input switches into the C(MQ). When a panel input switch is down it represents a 1; when it is up, it represents a zero.

Indicators. None.

Timing: 2 cycles

Execution. Since the address part of this instruction contains part of the operation code for this instruction, any address modification by an index register may result in the changing of the operation itself.

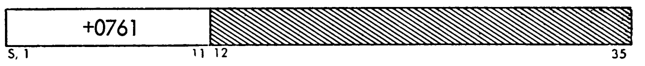
Control Instructions

Instructions which govern the flow of a program, and in particular those which cause an alteration in the computer's normal process of taking its instructions from sequential locations, are called *control instructions.*

Unconditional transfer instructions specify the location "Y" from which the computer is to take the next instruction. *Conditional transfer instructions* also specify a location Y. However, whether the computer takes its next instruction from Y or the next sequential location depends upon the outcome of a test. This test is specified by the operation code of the instruction.

Test instructions are similar to conditional control instructions in that they cause some test to be performed. Unlike conditional instructions, however, test instructions do not specify a location Y to which control may be transferred. Instead, the alternative location to which control may be transferred is fixed relative to the location of the test instruction.

NOP — No Operation

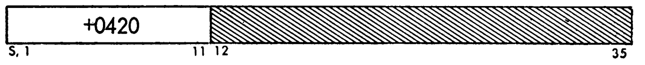


Description. This instruction causes the computer to take the next instruction in sequence.

Indicators. None.

Timing: 2 cycles

HPR — Halt and Proceed



Description. This instruction causes the computer to halt. The IC contains the location of the next se-

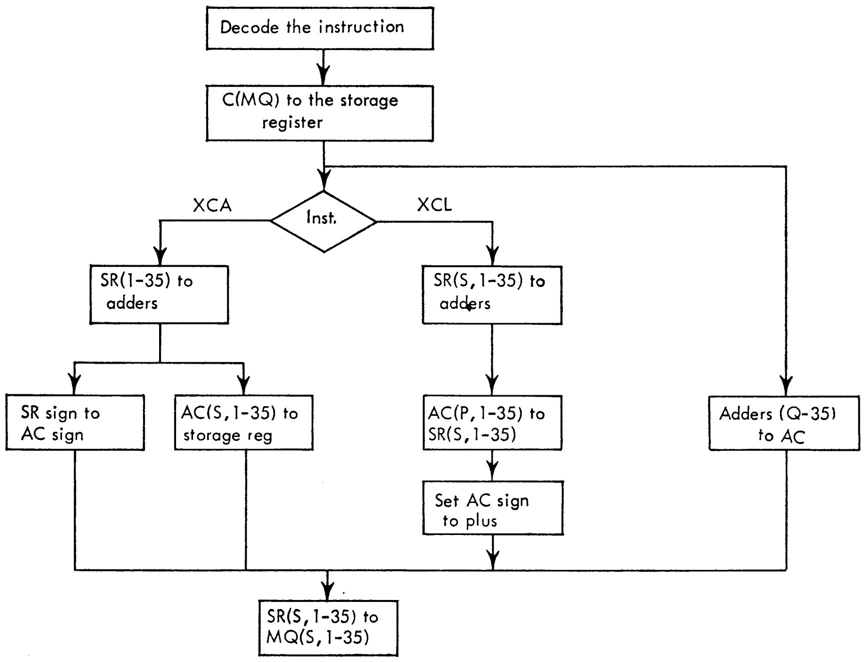


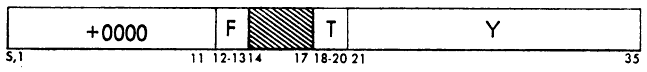
Figure 28. XCA and XCL Flow Chart

quential instruction. When the start key on the operator's console is depressed, the computer proceeds and executes the next sequential instruction.

Indicators. None.

Timing: 2 cycles

HTR — Halt and Transfer

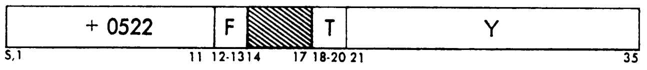


Description. This instruction causes the computer to halt. The IC contains the location of the HTR instruction. Depression of the start key, on the operator's console, causes the computer to transfer to location Y and execute that instruction.

Indicators. Trap Mode.

Timing: 2 cycles

XEC — Execute



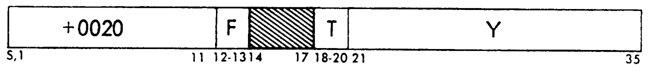
Description. This instruction causes the computer to perform or "execute" the instruction at location Y.

Indicators. None.

Timing: 1 cycle

Execution. Since the location counter is not altered (when Y contains any instruction other than a successful transfer or test instruction), the program advances to the next sequential instruction following the execute instruction after performing the instruction at location Y. If location Y contains a transfer instruction, it will be executed and program control will be altered from the sequential process. If location Y contains a test instruction, the instruction following EXECUTE will be located relative to the EXECUTE rather than the TEST instruction. Thus, any instruction which changes the instruction counter (STR, TRA, DCT, etc.) will alter program control when that instruction is executed by XEC.

TRA — Transfer



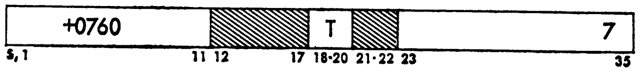
Description. This instruction causes the computer to take its next instruction from location Y and proceed from there.

Indicators. Trap Mode.

Timing: 1 cycle

Execution. See Figure 29.

ETM — Enter Trapping Mode



Description. This instruction causes the computer to enter the transfer trapping mode. The transfer trapping indicator on the operator's console is turned on.

Indicators. Trap Mode.

Timing: 2 cycles

Execution. When the computer is in the trapping mode and any transfer instruction except a TTR is executed, the location of the transfer instruction replaces the address part of location 0000 whether the condition for transferring is met or not. If the transfer condition is met, the computer takes its next instruction from location 0001 and proceeds from there. Only instructions which have "transfer" in their title are affected by the transfer trapping mode. Address

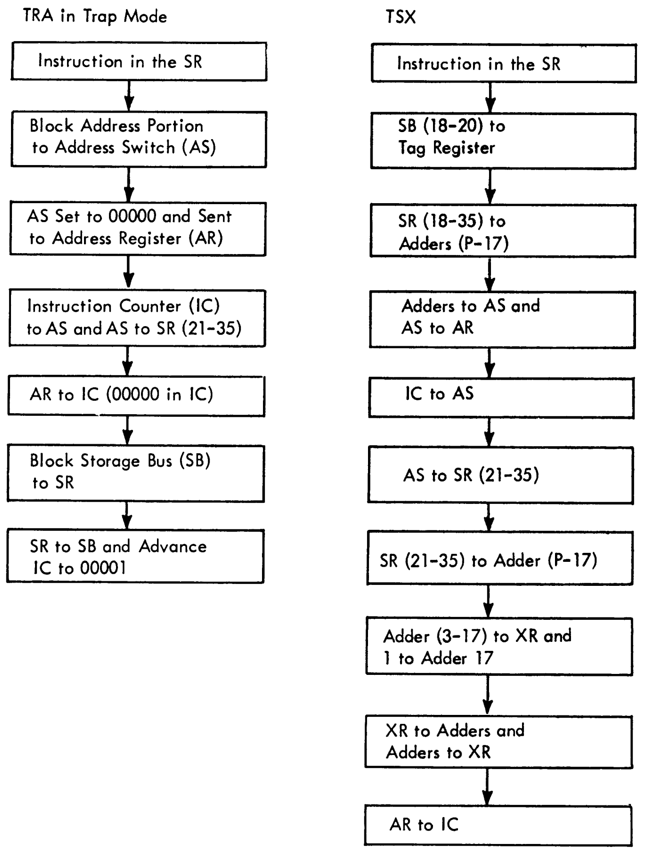

TRA in Trap Mode
Instruction in the SR
Block Address Portion to Address Switch (AS)
AS Set to 00000 and Sent to Address Register (AR)
Instruction Counter (IC) to AS and AS to SR (21-35)
AR to IC (00000 in IC)
Block Storage Bus (SB) to SR
SR to SB and Advance IC to 00001
TSX
Instruction in the SR
SB (18-20) to Tag Register
SR (18-35) to Adders (P-17)
Adders to AS and AS to AR
IC to AS
AS to SR (21-35)
SR (21-35) to Adder (P-17)
Adder (3-17) to XR and 1 to Adder 17
XR to Adders and Adders to XR
AR to IC

Figure 29. TRA, TSX, and Trap Mode Flow Chart

modification may change the operation, since positions 23-35 of the instruction are a part of the operation code.

LTM — Leave Trapping Mode

-0760	▨	T	▨	7
S,1 – 11	12 – 17	18-20	21-22	23 – 35

Description. This instruction turns off the trap mode indicator and causes the computer to leave the transfer trapping mode. Transfer instructions, therefore, will not be trapped again until an ETM operation is executed.

Indicators. Trap Mode.

Timing: 2 cycles

Execution. The computer operates in the trapping mode until either a LEAVE TRAPPING MODE OPERATION is executed or the clear or reset key on the operator's console is depressed. Since positions 23-35 represent part of the operation code of this instruction, any modification by an index register may result in the changing of the operation itself. See Figure 29.

TTR — Trap Transfer

+0021	F	▨	T	Y
S,1 – 11	12-13	14 – 17	18-20	21 – 35

Description. This instruction causes the computer to take its next instruction from location Y and to proceed from there whether in the transfer trap mode or not. This makes it possible to have an unconditional transfer in the transfer trapping mode.

Indicators. None.

Timing: 1 cycle

TZE — Transfer on Zero

+0100	F	▨	T	Y
S,1 – 11	12-13	14 – 17	18-20	21 – 35

Description. If the $C(AC)_{Q,P,1\text{-}35}$ are zero, the computer takes its next instruction from location Y and proceeds from there. If they are not zero, the next sequential instruction is taken.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 30.

TNZ — Transfer on No Zero

-0100	F	▨	T	Y
S,1 – 11	12-13	14 – 17	18-20	21 – 35

Description. If the $C(AC)_{Q,P,1\text{-}35}$ are not zero, the computer takes its next instruction from location Y and proceeds from there. If they are zero, the next sequential instruction is taken.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 30.

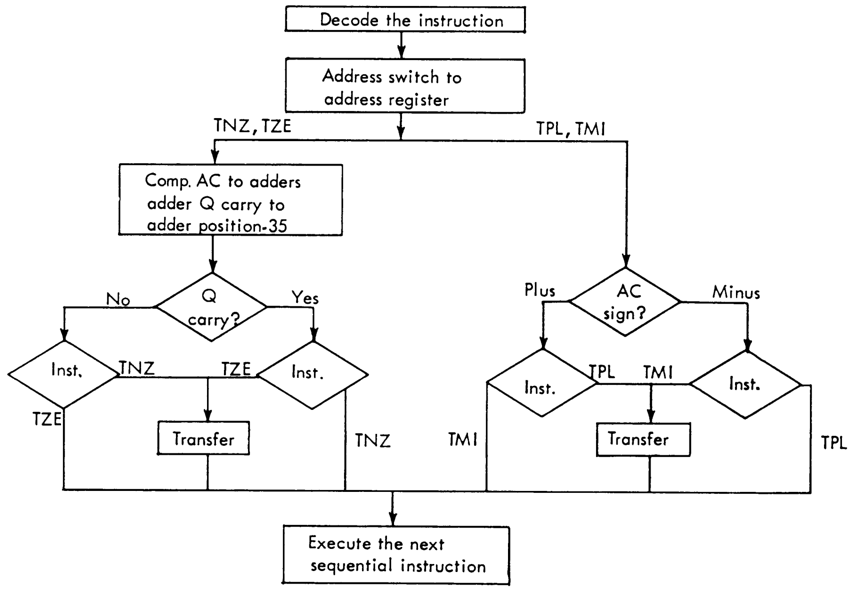


Figure 30. TNZ, TZE, TPL, and TMI Flow Chart

TPL — Transfer on Plus

+0120	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

Description. If the sign position of the AC is positive, the computer takes its next instruction from location Y and proceeds from there. If the sign position is negative, the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 1 cycle

Execution. See Figure 30.

TMI — Transfer on Minus

-0120	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

Description. If the sign position of the AC is negative, the computer takes its next instruction from location Y and proceeds from there. If the sign position is positive, the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 1 cycle

Execution. See Figure 30.

TOV — Transfer on Overflow

+0140	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

Description. If the AC overflow indicator is on, it is turned off and the computer takes its next instruction from location Y. If the indicator is off, the computer takes the next sequential instruction.

Indicators. AC overflow, trap mode.

Timing: 1 cycle

Execution. See Figure 31.

TNO — Transfer on No Overflow

-0140	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

Description. If the AC overflow indicator is off, the computer takes its next instruction from location Y. If the indicator is on, it is turned off and the computer takes the next sequential instruction.

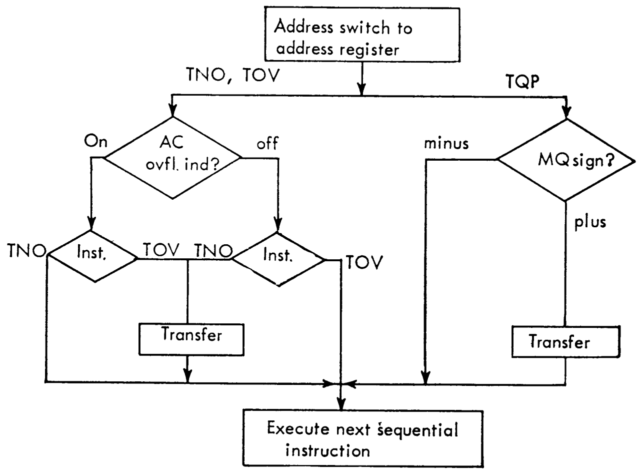


Figure 31. TOV, TNO, and TQO Flow Chart

Indicators. AC overflow, trap mode.

Timing: 1 cycle

Execution. See Figure 31.

TQP — Transfer on MQ Plus

+0162	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

Description. If the sign position of the MQ is plus, the computer takes its next instruction from location Y. If the sign position is negative, the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 1 cycle

Execution. See Figure 31.

TQO — Transfer on MQ Overflow

+0161	F	▨	T	Y
S,1 11	12-13	14 17	18-20	21 35

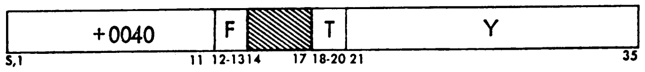
Description. This instruction is a conditional transfer when the computer is operating in the 704 floating-point mode. If the MQ overflow indicator is on, the computer takes its next instruction from location Y and turns the indicator off.

Indicators. MQ overflow.

Timing: 1 cycle

Execution. If this instruction is executed while the computer is in the normal mode, it is treated as a 1 cycle no-operation whether the MQ overflow indicator is on or not.

TLQ — Transfer on Low MQ



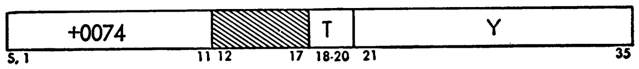
Description. If the c(MQ) are algebraically less than the c(AC), the computer takes its next instruction from location Y. If the c(MQ) are algebraically greater than or equal to the c(AC), the computer takes the next sequential instruction. NOTE: a plus zero is algebraically greater than a minus zero.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 32.

TSX — Transfer and Set Index



Description. This instruction places the 2's complement of the core address of the TSX(IC) in the specified index register (T). The computer takes its next instruction from location Y. NOTE: Subtracting the 2's complement of a number is equivalent to adding the number.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 29.

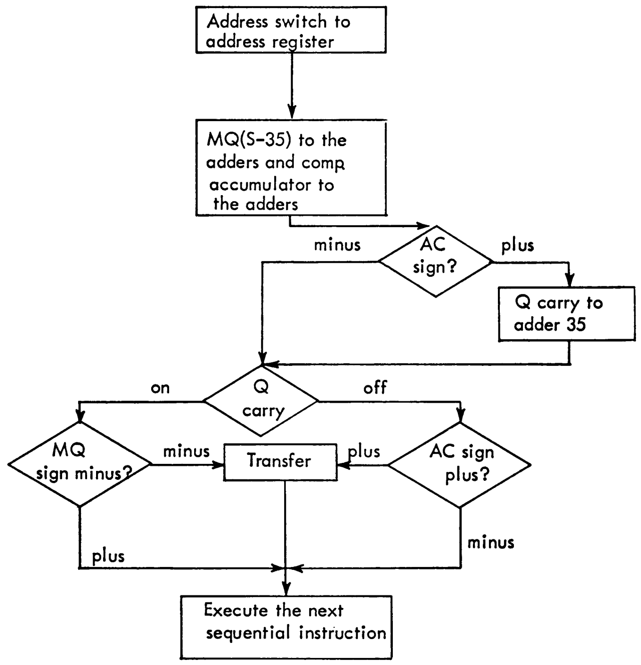
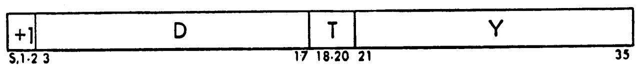

Address switch to address register
MQ(S-35) to the adders and comp. accumulator to the adders
AC sign?
minus
plus
Q carry to adder 35
Q carry
on
off
MQ sign minus?
minus
Transfer
plus
AC sign plus?
plus
minus
Execute the next sequential instruction

Figure 32. TLQ Flow Chart

TXI — Transfer with Index Incremented



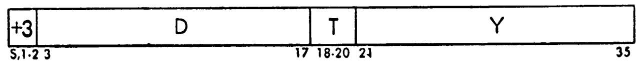
Description. This instruction adds the decrement (D) to the contents of the specified index register (T) and replaces the contents of the index register with the resulting sum. The computer then takes its next instruction from location Y.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 33.

TXH — Transfer on Index High



Description. If the number in the specified index register (T) is greater than the decrement (D), the computer takes its next instruction from location Y. If the number in the specified index register is less than or equal to D, the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 34.

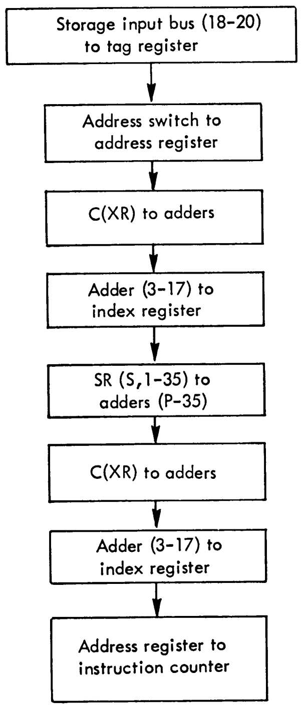

Storage input bus (18-20) to tag register
Address switch to address register
C(XR) to adders
Adder (3-17) to index register
SR (S,1-35) to adders (P-35)
C(XR) to adders
Adder (3-17) to index register
Address register to instruction counter

Figure 33. TXI Flow Chart

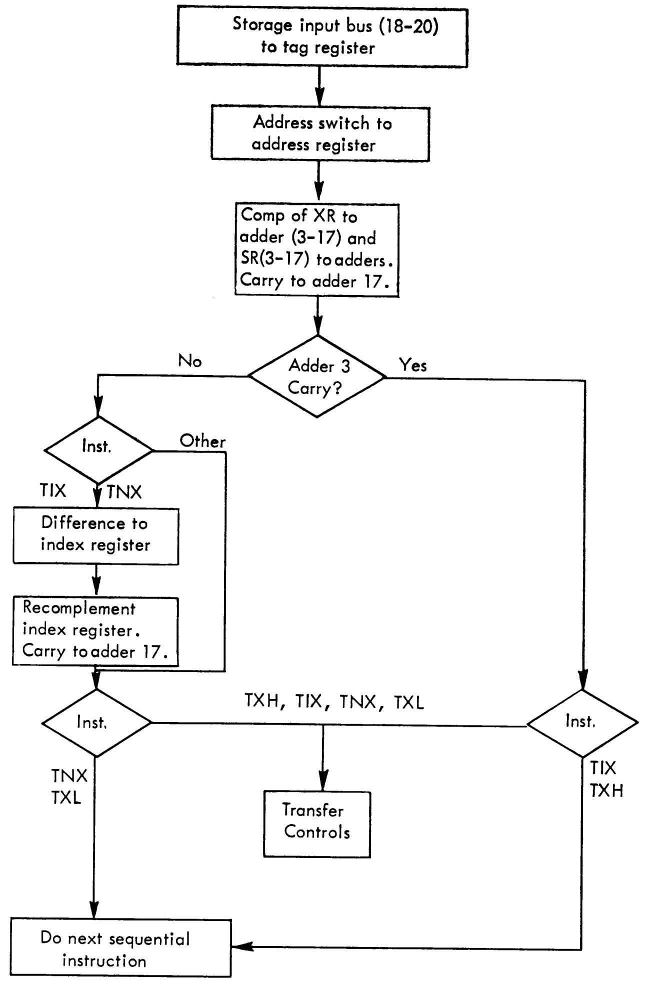
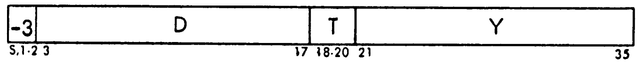

Storage input bus (18-20) to tag register
Address switch to address register
Comp of XR to adder (3-17) and SR(3-17) to adders. Carry to adder 17.
Adder 3 Carry?
No
Yes
Inst.
Other
TIX
TNX
Difference to index register
Recomplement index register. Carry to adder 17.
Inst.
TXH, TIX, TNX, TXL
Inst.
TNX
TXL
Transfer Controls
TIX
TXH
Do next sequential instruction

Figure 34. TIX, TXH, TNX, and TXL Flow Chart

TXL — Transfer on Index Low or Equal



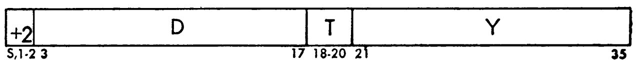
Description. If the contents of the index register specified by T are less than or equal to the D portion, the computer takes its next instruction from location Y. If the contents of T are greater than D, the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 34.

TIX — Transfer on Index



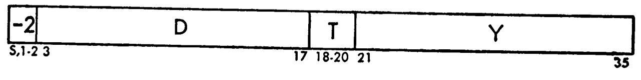
Description. If the C(XR) specified by T are greater than the C(D), the number in the index register is reduced by D and the computer takes its next instruction from Y. If C(T) is less than or equal to D, the C(T) are unchanged and the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 34.

TNX — Transfer on No Index



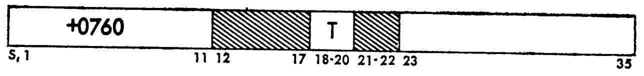
Description. If the C(XR) specified by T are equal to or less than D, the C(T) are unchanged and the computer takes its next instruction from Y. If C(T) are greater than D, the C(T) are reduced by D and the computer takes the next sequential instruction.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 34.

PSE — Plus Sense



Description. This instruction provides a means of testing the status of the sense switches and of turning on or off the sense lights on the operator's console. The instruction also permits the transmission of an impulse to or from the exit or entry hubs on the printer or card punch control panels. Address modification may cause the operation code to be changed.

Indicators. Sense indicators and switches.

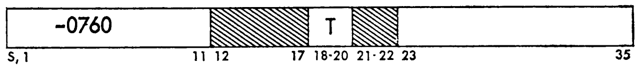
Timing: 2 cycles

Execution. The address part (23-35) of this instruction determines whether a light, switch, printer, or card punch is being sensed. Further, it determines which light, switch, or hub is sensed. The octal addresses for sense instructions are:

SLF 0140	Turns off all sense lights on the operator's console.
SLN 0141- 0144	Turn on sense lights 1, 2, 3 or 4, respectively, on the console.
SWT 0161- 0166	Test sense switches on the console. If the corresponding switch is down (on), the computer skips the next instruction and proceeds from there. If the sense switch is up (off), the computer takes the next sequential instruction.

SPU	
1341- 1342 (A) 2341- 2342 (B) 3341- 3342 (C) 4341- 4342 (D) 5341- 5342 (E) 6341- 6342 (F) 7341- 7342 (G) 10341-10342 (H)	An impulse will appear at the specified exit hub of the card punch control panel attached to appropriate data channel. Hubs are numbered 1 and 2.
SPT	
1360 (A) 2360 (B) 3360 (C) 4360 (D) 5360 (E) 6360 (F) 7360 (G) 10360 (H)	If an impulse is present at the sense entry hub of the printer control panel, the computer skips the next instruction and proceeds from there. If no impulse is present, the computer takes the next sequential instruction.
SPR	
1361- 1372 (A) 2361- 2372 (B) 3361- 3372 (C) 4361- 4372 (D) 5361- 5372 (E) 6361- 6372 (F) 7361- 7372 (G) 10361-10372 (H)	The computer causes an impulse to appear at the specified hub on the control panel of the printer attached to that particular data channel.

MSE — Minus Sense



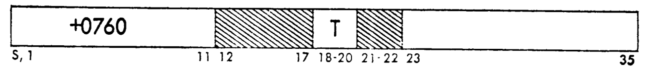
Description. This instruction provides a means of testing the status of the sense lights on the operator's console. The lights may be turned on by a PSE instruction with an address of 0141 to 0144. Address modification may cause the operation code to be changed.

Indicators. Sense lights.

Timing: 2 cycles

Execution. The addresses of the four sense lights are 0141 to 0144. If the corresponding sense light is on, the light is turned off and the computer skips the next instruction and proceeds from there. If the light is off, the computer executes the next sequential instruction.

BTT — Beginning of Tape Test



Description. This instruction is used to test the status of data channel beginning-of-tape indicators. The channel whose indicator is to be tested is specified by the address portion (Y) of the BTT instruction. Address modification may cause the operation code to be changed. The addresses for the channels are:

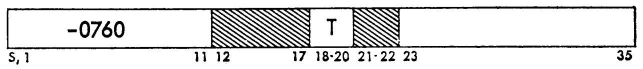
Data channel A	1000
Data channel B	2000
Data channel C	3000
Data channel D	4000
Data channel E	5000
Data channel F	6000
Data channel G	7000
Data channel H	10000

Indicators. All beginning-of-tape indicators.

Timing: 2 cycles

Execution. If the beginning-of-tape indicator for data channel Y is on, the computer takes the next sequential instruction, and the indicator is turned off. If the beginning-of-tape indicator is off, the computer skips the next instruction and proceeds from there. The beginning-of-tape indicator is turned on by a backspace record or backspace file instruction given to a tape unit that is positioned at its load point. See Figure 35.

ETT — End of Tape Test



Description. This instruction is used to test the status of data channel end-of-tape indicators. The channel whose indicator is to be tested is specified by the address portion of the ETT instruction. The addressing system is the same as specified for the BTT instruction. Address modification may cause the operation code to be changed.

Indicators. End-of-tape indicators.

Timing: 2 cycles

Execution. If the end-of-tape indicator for data channel Y is on, the computer takes the next sequential instruction and turns the indicator off. If the

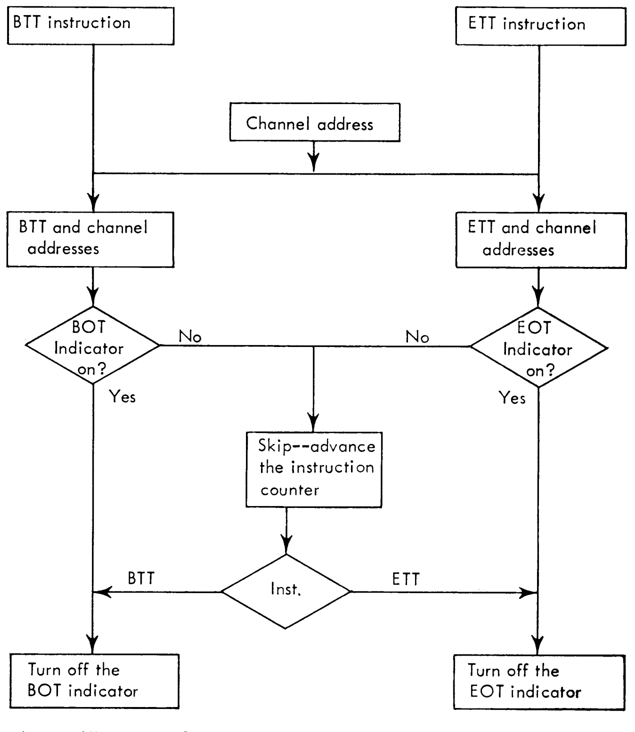

BTT instruction
ETT instruction
Channel address
BTT and channel addresses
ETT and channel addresses
BOT Indicator on?
No
No
EOT Indicator on?
Yes
Yes
Skip--advance the instruction counter
BTT
Inst.
ETT
Turn off the BOT indicator
Turn off the EOT indicator

Figure 35. BTT and ETT Flow Chart

indicator is off, the computer skips the next instruction and proceeds from there. The end-of-tape indicator is turned on when either a write select or a write end of file causes the end-of-tape marker to be passed over. See Figure 35.

IOT — Input-Output Check Test

+0760	▨	T	▨	5
S, 1	11 12	17 18-20	21-22 23	35

Description. If the I-O check indicator is on, the indicator is turned off and the computer takes the next sequential instruction. If the indicator is off, the computer skips the next instruction and proceeds from there. Any address modification may result in the changing of the operation itself.

Indicators. I-O check.

Timing: 2 cycles

Execution. The I-O check indicator will be turned on by any of the following conditions:

1. If a write CRT or drum select instruction is executed and the computer is not in the select trap mode.

2. If COPY, COPY AND ADD LOGICAL, or LOCATE DRUM ADDRESS are executed when the computer is not in the copy trap mode.

3. If a RESET AND LOAD CHANNEL or a LOAD CHANNEL is executed and the specified channel is not selected.

4. If, when writing, a channel data register has not been loaded with a word from storage by the time its contents are to be sent to the output unit.

5. If, when reading, a channel data register has not stored its contents by the time new data are to be sent from an input unit.

PBT — P-Bit Test

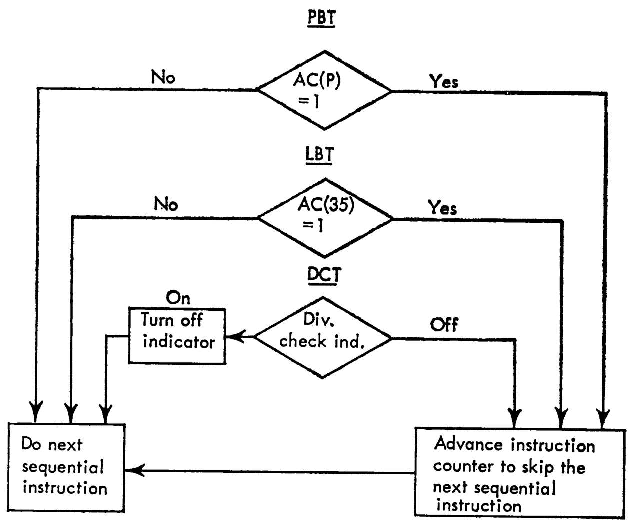
-0760	▨	T	▨	1
S, 1	11 12	17 18-20	21-22 23	35

Description. If the C (AC) $_P$ is a 1, the computer skips the next instruction and proceeds from there. If P contains a 0, the computer takes the next sequential instruction. Address modification may result in the changing of the instruction itself.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 36.



PBT
No
AC(P) =1
Yes
LBT
No
AC(35) =1
Yes
DCT
On
Turn off indicator
Div. check ind.
Off
Do next sequential instruction
Advance instruction counter to skip the next sequential instruction

Figure 36. PBT, LBT, and DCT Flow Chart

LBT — Low-Order Bit Test

+0760	▨	T	▨	1
S, 1	11 12	17 18-20	21-22 23	35

Description. If the C(AC)$_{35}$ is a 1, the computer skips the next instruction and proceeds from there. If 35 is a 0, the computer takes the next sequential instruction. Address modification may result in the changing of the instruction.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 36.

DCT — Divide Check Test

+0760	▨	T	▨	12
S, 1	11 12	17 18-20	21-22 23	35

Description. If the indicator is on, it is turned off and the computer takes the next sequential instruction. If the indicator is off, the computer skips the next instruction and proceeds from there. Address modification may result in the changing of the instruction itself.

Indicators. Divide check.

Timing: 2 cycles

Execution. See Figure 36.

ZET — Storage Zero Test

+0520	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. If the $C(Y)_{1\text{-}35}$ are 0, the computer skips the next instruction and proceeds from there. If the $C(Y)_{1\text{-}35}$ are not zero, the computer takes the next sequential instruction. The C(Y) are not changed.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 37.

NZT — Storage not Zero Test

-0520	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. If the $C(Y)_{1\text{-}35}$ are not 0, the computer skips the next instruction and proceeds from there. If the $C(Y)_{1\text{-}35}$ are 0, the computer takes the next sequential instruction. The C(Y) are unchanged.

Indicators. None.

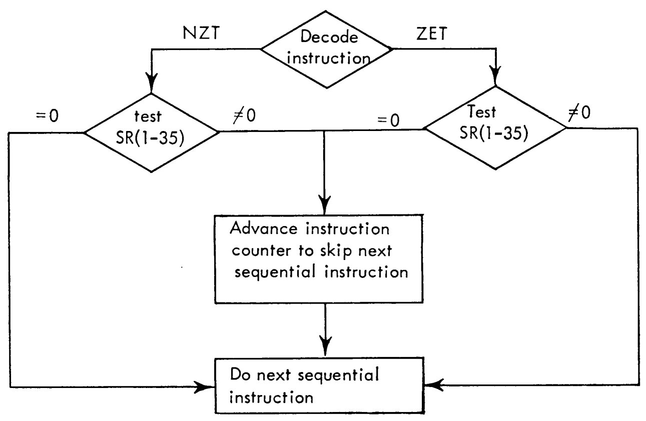
Timing: 2 cycles

Execution. See Figure 37.

CAS — Compare Accumulator with Storage

+0340	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. If the C(AC) are algebraically greater than the C(Y), the computer takes the next sequential instruction. If the C(AC) are algebraically equal to the C(Y), the computer skips the next instruction and proceeds from there. If the C(AC) are algebraically less than the C(Y) the computer skips the next two instructions and proceeds from there.



NZT
Decode instruction
ZET
=0
test SR(1-35)
≠0
=0
Test SR(1-35)
≠0
Advance instruction counter to skip next sequential instruction
Do next sequential instruction

Figure 37. NZT and ZET Flow Chart

Indicators. None.

Timing: 3 cycles

Execution. NOTE: Two numbers are considered algebraically equal if the magnitudes and signs of both are equal. A plus zero is algebraically greater than a minus zero.

LAS — Logical Compare Accumulator with Storage

-0340	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

Description. The $C(AC)_{Q,P,1\text{-}35}$ are treated as an unsigned 37-bit number and are compared with the $C(Y)_{S,1\text{-}35}$ which are treated as an unsigned 36-bit quantity. If the $C(AC)_{Q,P,1\text{-}35}$ are greater than the C(Y), the computer takes the next sequential instruction. If the $C(AC)_{Q,P,1\text{-}35}$ are equal to the C(Y), the computer skips the next instruction and proceeds from there. If the $C(AC)_{Q,P,1\text{-}35}$ are less than the C(Y), the computer skips the next two instructions and proceeds from there.

Indicators. None.

Timing: 3 cycles

FOR THE FOLLOWING control operations that refer to data channels, the description of the operation is given for channel A. For the other channels, the operation code and the title are given.

TCOA — Transfer on Channel A in Operation

+0060	F	▨	T	Y
S,1 … 11	12-13	14 … 17	18-20	21 … 35

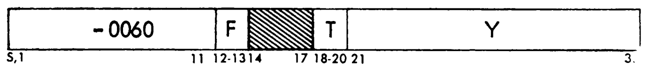
Description. If channel A is in operation, the computer takes its next instruction from location Y. If the channel is not in operation, the computer takes the next sequential instruction, and the operation of the channel is not affected. The channel is in operation as long as a select register contains information.

Indicators. Trap.

Timing: 2 cycles

INSTRUCTION	CODE	NAME
TCOB	+0061	Transfer on Channel B in Operation
TCOC	+0062	Transfer on Channel C in Operation
TCOD	+0063	Transfer on Channel D in Operation
TCOE	+0064	Transfer on Channel E in Operation
TCOF	+0065	Transfer on Channel F in Operation
TCOG	+0066	Transfer on Channel G in Operation
TCOH	+0067	Transfer on Channel H in Operation

TCNA — Transfer on Channel A not in Operation



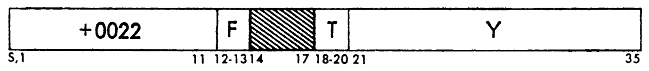
Description. If channel A is not in operation, the computer takes its next instruction from location Y. If the channel is in operation, the computer takes the next sequential instruction, and the operation of the channel is not affected.

Indicators. Trap.

Timing: 2 cycles.

INSTRUCTION	CODE	NAME
TCNB	−0061	Transfer on Channel B not in Operation
TCNC	−0062	Transfer on Channel C not in Operation
TCND	−0063	Transfer on Channel D not in Operation
TCNE	−0064	Transfer on Channel E not in Operation
TCNF	−0065	Transfer on Channel F not in Operation
TCNG	−0066	Transfer on Channel G not in Operation
TCNH	−0067	Transfer on Channel H not in Operation

TRCA — Transfer on Channel A Redundancy Check



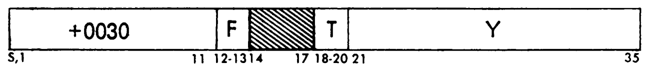
Description. If the tape check indicator for channel A is on, it is turned off and the computer takes its next instruction from location Y. If the indicator is off, the next sequential instruction is taken.

Indicators. Tape Check, Trap.

Timing: 2 cycles.

INSTRUCTION	CODE	NAME
TRCB	−0022	Transfer on Channel B Redundancy Check
TRCC	+0024	Transfer on Channel C Redundancy Check
TRCD	−0024	Transfer on Channel D Redundancy Check
TRCE	+0026	Transfer on Channel E Redundancy Check
TRCF	−0026	Transfer on Channel F Redundancy Check
TRCG	+0027	Transfer on Channel G Redundancy Check
TRCH	−0027	Transfer on Channel H Redundancy Check

TEFA — Transfer on Channel A End of File



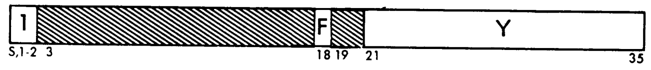
Description. If the end-of-file indicator for channel A is on, it is turned off and the computer takes its next instruction from location Y. If the indicator is off, the computer takes the next sequential instruction.

Indicators. End-of-file, Trap.

Timing: 2 cycles.

INSTRUCTION	CODE	NAME
TEFB	−0030	Transfer on Channel B End of File
TEFC	+0031	Transfer on Channel C End of File
TEFD	−0031	Transfer on Channel D End of File
TEFE	+0032	Transfer on Channel E End of File
TEFF	−0032	Transfer on Channel F End of File
TEFG	+0033	Transfer on Channel G End of File
TEFH	−0033	Transfer on Channel H End of File

TCH — Transfer in Channel



Description. This command is the transfer command for all data channels.

Indicators. None.

Timing: 2 cycles

Execution. When a TCH command is executed, the data channel proceeds immediately to its next command which is taken from location Y. The location register is set to Y + 1. The command located at Y is then loaded into the data channel.

Index Transmission Operations

This section of operations deals with the loading and storing of the contents of index registers.

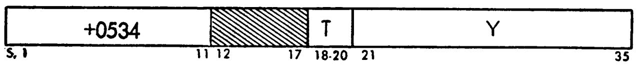
The operations always involve one or more index registers and either the address or decrement field of some location in storage or the accumulator register. The following 15-bit fields may serve as one of the agents in an index transmission operation: the address or decrement of the accumulator, the address or decrement of any location in storage, or the address part of the index transmission instruction itself. In addition, the number to be loaded may be placed in the specified index register in either true or complement form.

Single registers or any combination of index registers may be specified. If more than one register is specified in an unloading operation, their contents are "OR'ed" together to produce the effective number. OR'ing matches the registers position-for-position. If there is a bit in either or both of the registers, the result is a bit. For example:

```
      XRA 101100
      XRB 011000
    ------------
   Result 111100
```

If more than one index register is specified in a loading operation, the data are loaded into all registers specified.

LXA — Load Index from Address



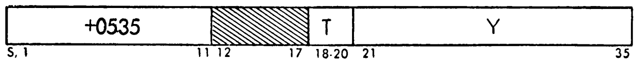
Description. The C(Y)$_{21-35}$ replace the contents of the specified index register. The C(Y) are unchanged

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 38.

LAC — Load Complement of Address in Index



Description. The 2's complement of the C(Y)$_{21-35}$ replaces the contents of the specified index register. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 38.

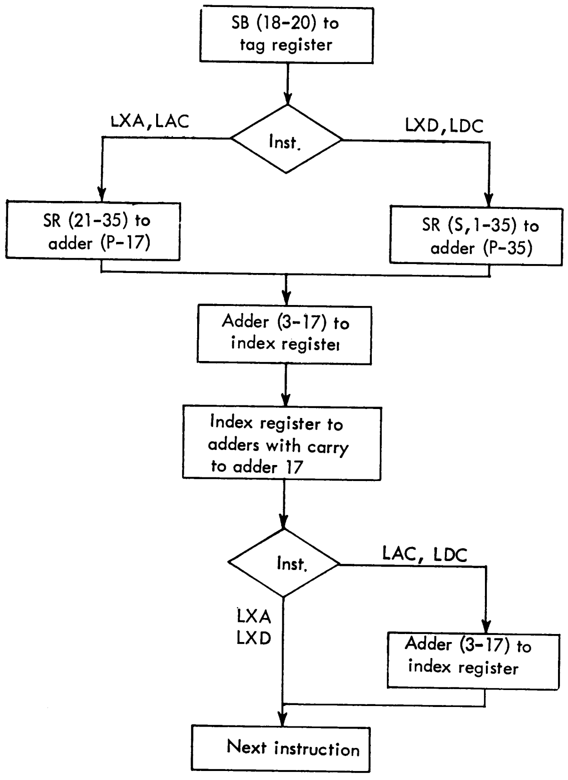
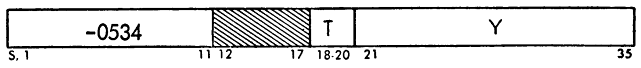

SB (18-20) to tag register
Inst.
LXA, LAC
LXD, LDC
SR (21-35) to adder (P-17)
SR (S, 1-35) to adder (P-35)
Adder (3-17) to index registei
Index register to adders with carry to adder 17
Inst.
LAC, LDC
LXA LXD
Adder (3-17) to index register
Next instruction

Figure 38. LXA, LAC, LXD, and LDC Flow Chart

LXD — Load Index from Decrement



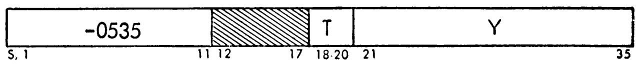
Description. The C(Y)$_{3-17}$ replace the contents of the specified index register. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 38.

LDC — Load Complement of Decrement in Index



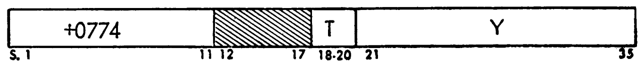
Description. The 2's complement of the C(Y)$_{3-17}$ replaces the contents of the specified index register. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 38.

AXT — Address to Index True



Description. Positions 21-35 of this instruction replace the contents of the specified index register. The instruction is unchanged.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 39.

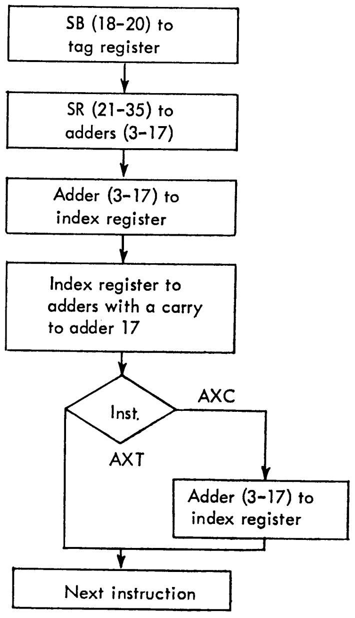
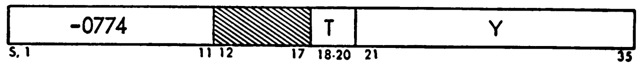

SB (18-20) to tag register
SR (21-35) to adders (3-17)
Adder (3-17) to index register
Index register to adders with a carry to adder 17
Inst.
AXC
AXT
Adder (3-17) to index register
Next instruction

Figure 39. AXT and AXC Flow Chart

AXC — Address to Index Complemented



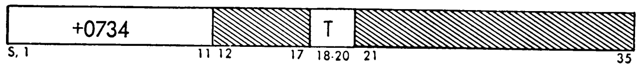
Description. The 2's complement of positions 21-35 of this instruction replaces the contents of the specified index register. The instruction is unchanged.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 39.

PAX — Place Address in Index



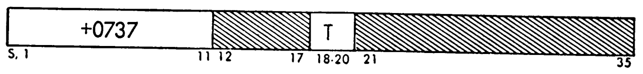
Description. The c(AC)$_{21-35}$ replace the contents of the specified index register. The c(AC) are unchanged.

Indicators. None.

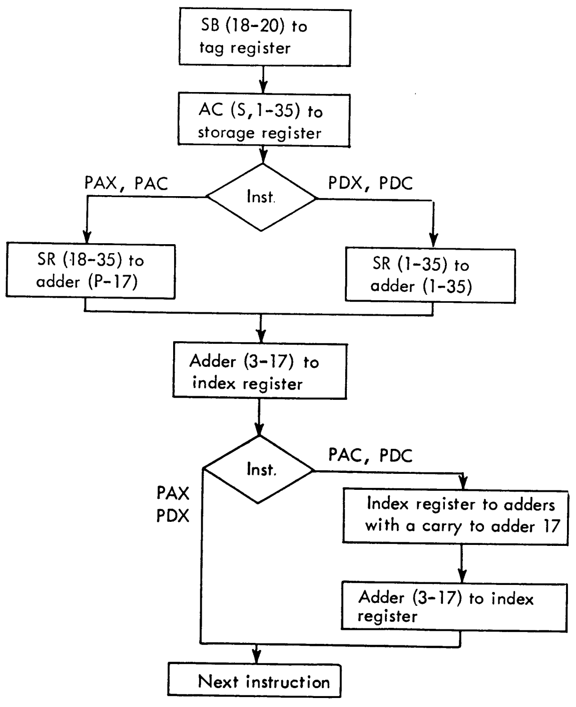
Timing: 1 cycle

Execution. See Figure 40.

PAC — Place Complement of Address in Index



Description. The 2's complement of the c(AC)$_{21-35}$ replaces the contents of the specified index register. The c(AC) are unchanged.



SB (18-20) to tag register
AC (S,1-35) to storage register
PAX, PAC
Inst.
PDX, PDC
SR (18-35) to adder (P-17)
SR (1-35) to adder (1-35)
Adder (3-17) to index register
Inst.
PAC, PDC
PAX PDX
Index register to adders with a carry to adder 17
Adder (3-17) to index register
Next instruction

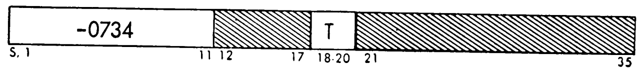
Figure 40. PAX, PAC, PDX, and PDC Flow Chart

Indicators. None.

Timing: 1 cycle

Execution. See Figure 40.

PDX — Place Decrement in Index



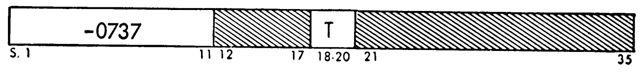
Description. The c(AC)$_{3-17}$ replace the contents of the specified index register. The c(AC) are unchanged.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 40.

PDC — Place Complement of Decrement in Index



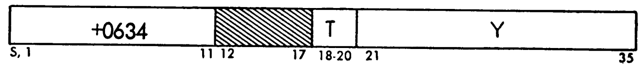
Description. The 2's complement of the c(AC)$_{3-17}$ replaces the contents of the specified index register. The c(AC) are unchanged.

Indicators. None.

Timing: 1 cycle

Execution. See Figure 40.

SXA — Store Index in Address



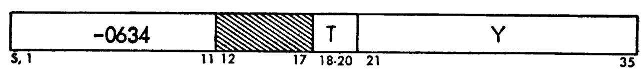
Description. The c(Y)$_{21-35}$ are replaced by the contents of the specified index register. The c(Y)$_{S,1-20}$ are unchanged. With a tag of 0, c(Y)$_{21-35}$ are replaced with zeros.

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 41.

SXD — Store Index in Decrement



Description. The c(Y)$_{3-17}$ are replaced by the contents of the specified index register. The c(Y)$_{S,1,2,18-35}$ are unchanged. With a tag of 0, decrement is replaced with zeros.

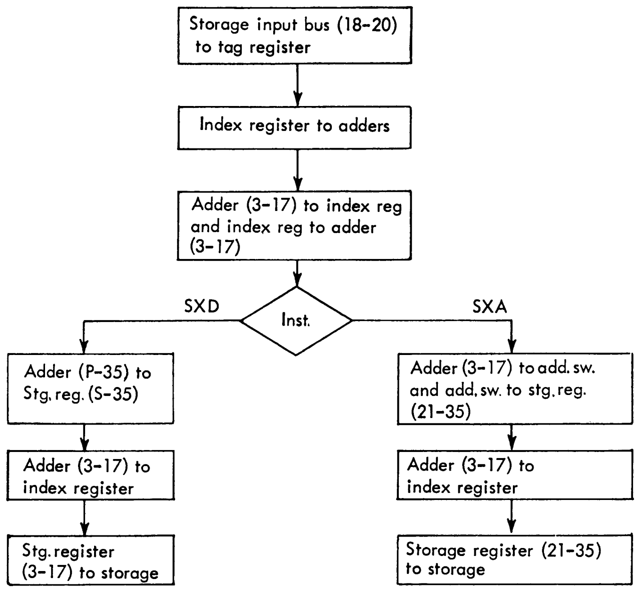

Storage input bus (18-20) to tag register
Index register to adders
Adder (3-17) to index reg and index reg to adder (3-17)
SXD
Inst.
SXA
Adder (P-35) to Stg. reg. (S-35)
Adder (3-17) to index register
Stg. register (3-17) to storage
Adder (3-17) to add. sw. and add. sw. to stg. reg. (21-35)
Adder (3-17) to index register
Storage register (21-35) to storage

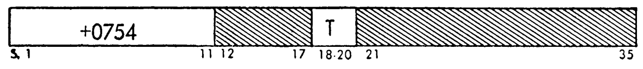
Figure 41. SXA and SXD Flow Chart

Indicators. None.

Timing: 2 cycles.

Execution. See Figure 41.

PXA — Place Index in Address



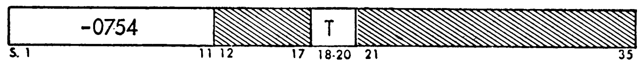
Description. The entire accumulator is cleared and the contents of the specified index register are placed in the address part of the AC_{21-35}. With a tag of 0 the C (AC) are set to zeros. XR is unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 42.

PXD — Place Index in Decrement



Description. The entire accumulator is cleared and the contents of the specified index register are placed in the decrement part of the AC_{3-17}. With a tag of 0, the C (AC) are set to zeros. XR is unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 42.

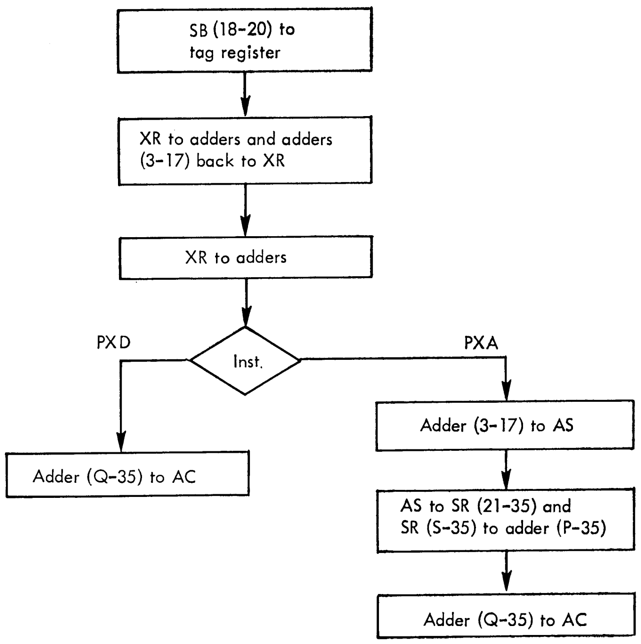

SB (18-20) to tag register
XR to adders and adders (3-17) back to XR
XR to adders
PXD
Inst.
PXA
Adder (Q-35) to AC
Adder (3-17) to AS
AS to SR (21-35) and SR (S-35) to adder (P-35)
Adder (Q-35) to AC

Figure 42. PXD and PXA Flow Chart

Logical Operations

Logical instructions operate on a 36-bit word. The sign position is simply another bit position. The exception to this is when the P position is used instead of the sign position. Logical instructions are frequently used in a process called *masking*. This is the process of extracting one or more small parts of a word from the whole word.

The AND and OR concept is used with logical operations. When two numbers are combined by an AND, they are matched bit-for-bit. If the same position in each word contains a 1, the result is a 1. If in one word the position is 0 and in the other word it is a 1, the result is a 0. If the same position in both words is a zero, the result is a 0. The following is an example of a logical AND operation:

```
101101011011
101001001101
------------
101001001001 Resulting AND
```

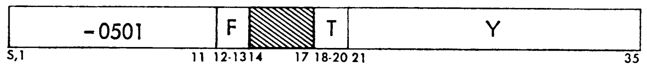
An OR function (sometimes called "inclusive OR") also matches two numbers bit-for-bit. The difference, however, when compared with an AND, is: (1) if the same position in either word contains a 1, the result is a 1; (2) if the same position in both words is a 1, the result is again a 1; (3) only if the same position in both words is a 0, is the resulting position a 0. For example:

```
011010110101
001100100100
------------
011110110101 Resulting inclusive OR
```

One other function of the logical operations is an *exclusive* OR. In this operation, only those positions which do not match result in a 1. If the same position in each word is a zero or if the same position in each word is a 1, the result is a zero. If the same position in one word is a 1 and in the other a 0, then the result is a 1. For example:

```
101101100101
001011001101
────────────
100110101000 Resulting exclusive OR
```

ORA — OR to Accumulator



Description. Each bit of the c(Y)$_{S,1-35}$ is matched with the corresponding bit of the c(AC)$_{P,1-35}$. c(Y)$_S$ is matched with c(AC)$_P$.

Indicators. None.

Timing: 2 cycles

Execution. When the corresponding bit of either location Y or the AC (or both) is a 1, a 1 replaces the contents of that position in the AC. When the corresponding bits of both location Y and the AC are 0's, a 0 replaces the contents of that position of the AC. The c(Y) and the S and Q positions of the AC are unchanged. See Figure 43.

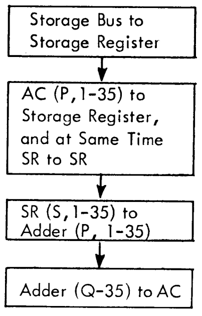
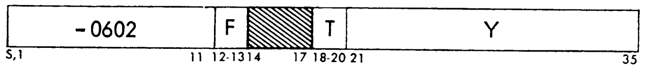

Storage Bus to Storage Register
AC (P,1-35) to Storage Register, and at Same Time SR to SR
SR (S,1-35) to Adder (P, 1-35)
Adder (Q-35) to AC

Figure 43. ORA Flow Chart

ORS — OR to Storage



Description. Each bit of the c(AC)$_{P,1-35}$ is matched with the corresponding bit of the c(Y)$_{S,1-35}$, c(AC)$_P$ being matched with c(Y)$_S$.

Indicators. None.

Timing: 2 cycles

Execution. When the corresponding bit of either the AC or location Y (or both) is a 1, a 1 replaces the contents of that position in location Y. When the corresponding bits of both the AC and location Y are 0's, a 0 replaces the contents of that position in location Y. The c(AC) are unchanged. See Figure 44.

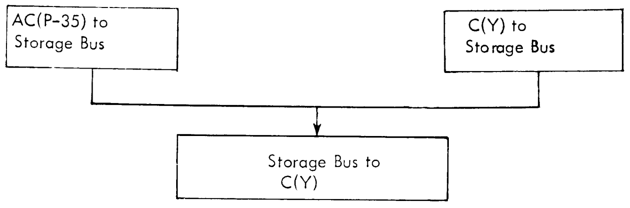
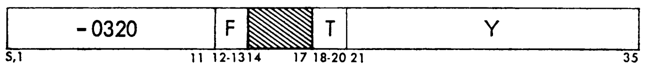


Figure 44. ORS Flow Chart

ANA — AND to Accumulator



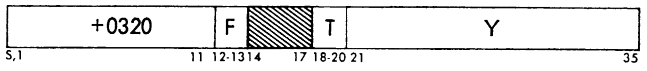
Description. Each bit of the c(Y)$_{S,1-35}$ is matched with the corresponding bit of the c(AC)$_{P,1-35}$. c(Y)$_S$ is matched with c(AC)$_P$.

Indicators. None.

Timing: 3 cycles

Execution. When the corresponding bits of both location Y and the AC are 1's, a 1 replaces the contents of that position in the AC. When the corresponding bit of either location Y or the AC, or both, is a 0, a 0 replaces the contents of that position in the AC. The S and Q positions of the AC are cleared. The c(Y) are unchanged. See Figure 45.

ANS — AND to Storage



Description. Each bit of the c(AC)$_{P,1-35}$ is matched with the corresponding bit of the c(Y)$_{S,1-35}$, c(AC)$_P$ being matched with c(Y)$_S$.

Indicators. None.

Timing: 4 cycles

Execution. When the corresponding bits of both the AC and location Y are 1's, a 1 replaces the contents of that position in location Y. When the corresponding bit of either the AC or location Y, or both, is a 0, a 0 replaces the contents of that position in location Y. The c(AC) are unchanged. See Figure 45.

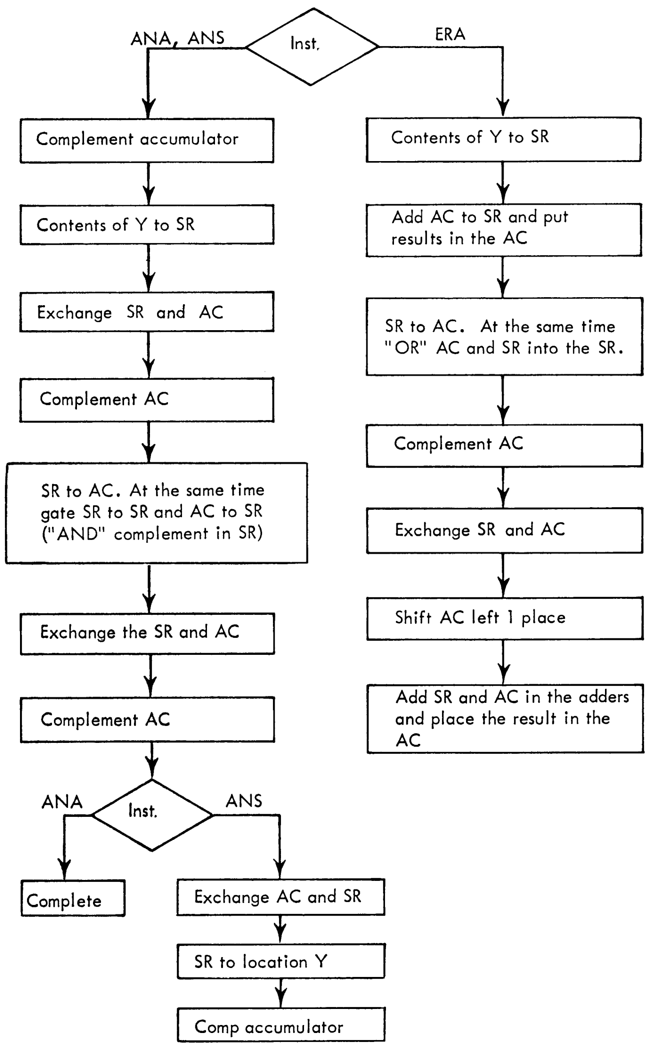
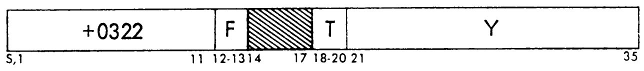


Figure 45. ANA, ANS, and ERA Flow Chart

ERA — Exclusive OR to Accumulator



Description. Each bit of the $C(Y)_{S,1\text{-}35}$ is matched with the corresponding bit of the $C(AC)_{P,1\text{-}35}$, $C(Y)_S$ being matched with $C(AC)_P$.

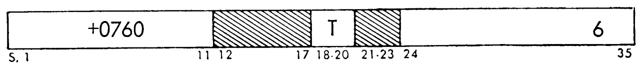
Indicators. None.

Timing: 3 cycles

Execution. When the corresponding position of the AC matches the position in location Y, a 0 replaces the contents of that position in the AC. When the corresponding position of the AC does not match the position in location Y, a 1 replaces the contents of that position in the AC. Positions S and Q of the AC are cleared. The C(Y) are unchanged. See Figure 45.

THE FOLLOWING logical operations affect the contents of the accumulator only.

COM — Complement Magnitude



Description. All 1's are replaced by 0's and all 0's are replaced by 1's in the $C(AC)_{Q,P,1\text{-}35}$. The $C(AC)_S$ is unchanged. Address modification may change the instruction itself.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 46.

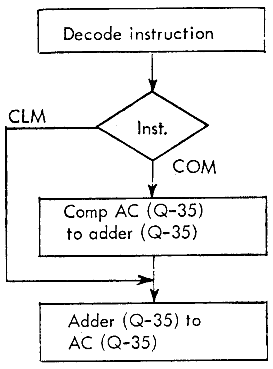
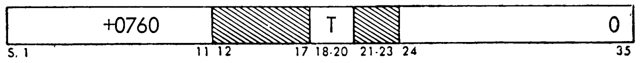


Figure 46. CLM and COM Flow Chart

CLM — Clear Magnitude



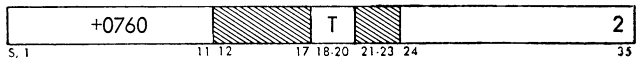
Description. The $C(AC)_{Q,P,1\text{-}35}$ are cleared. The $C(AC)_S$ are unchanged. Address modification by an index register may change the operation itself.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 46.

CHS — Change Sign



Description. If AC sign is plus, it is made negative. If it is negative, it is made plus. Address modification by an index register may change the operation itself. $C(AC)_{Q,P,1\text{-}35}$ are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 47.

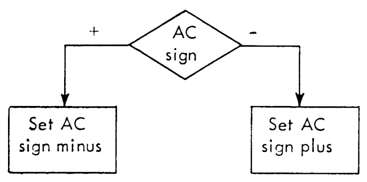
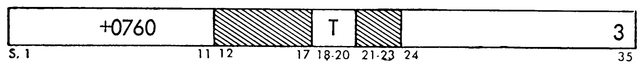


Figure 47. CHS Flow Chart

SSP — Set Sign Plus



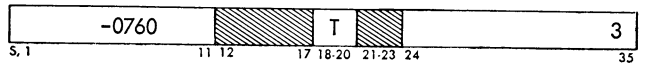
Description. The sign of the AC is set to plus (0). Address modification by an index register may result in changing the operation itself.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 48.

SSM — Set Sign Minus



Description. The sign of the AC is set to minus (1). Address modification may result in changing the operation itself.

Indicators. None

Timing: 2 cycles

Execution. See Figure 48.

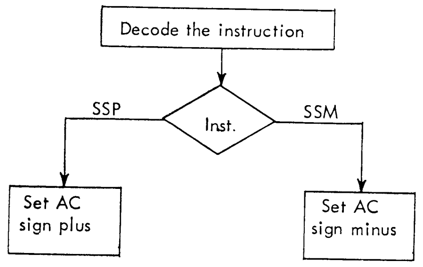


Figure 48. SSM and SSP Flow Chart

Sense Indicator Operations

The following 24 instructions make reference to the 36-bit sense indicator (SI) register. The 36 bits of the SI may be thought of as switches which may be turned on or off and tested either singly or in groups by the program.

The contents of the SI register are manipulated through the use of a *mask*. The mask is a bit pattern comprised of 1's and 0's which may appear in an instruction, the AC, or any storage location. All masks for SI operations are used in the same way; that is, each position in the mask is compared with the corresponding position in the SI register. For the positions in the mask which contain a 1, the corresponding position in the SI is either modified or tested depending upon the SI operation used. For the zero bits in the mask, the corresponding positions of the SI are not affected.

Four of the sense indicator operations are concerned with the transmission of full 36-bit words between the SI and either the AC or core storage. The remaining 20 operations are used to test or modify the C(SI). These 20 operations may be classified by the following five functions:

1. *Set or Logical OR.* These operations replace with a 1, the contents of each SI position selected by the mask.

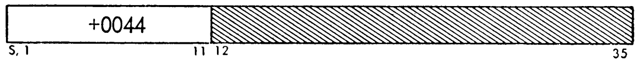
2. *Reset.* These operations replace with a 0 the contents of each SI position selected by the mask.

3. *Invert.* These operations replace the contents of each SI position selected by the mask with its complement; i.e., 1's are replaced by 0's and 0's are replaced by 1's.

4. *On Test.* These operations examine the contents of each SI position selected by the mask. If all examined positions contain a 1, the calculator will either transfer to a location Y or skip the next instruction, depending upon the testing operation used.

5. *Off Test.* These operations examine the contents of each SI position selected by the mask. If all examined positions contain a 0, the calculator either transfers to location Y or skips the next instruction, depending upon the testing operation used.

PAI — Place Accumulator in Indicators



Description. The C (AC)$_{P,1\text{-}35}$ replace the C (SI)$_{0\text{-}35}$. The C (AC) are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 49.

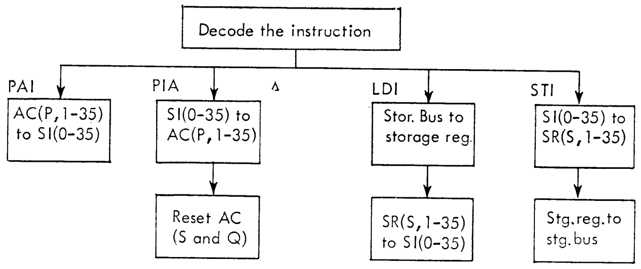
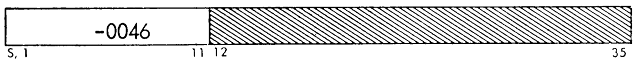


Figure 49. PAI, PIA, LDI, and STI Flow Chart

PIA — Place Indicators in Accumulator



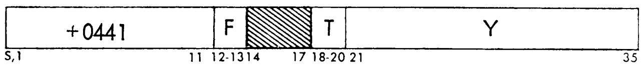
Description. The $C(SI)_{0-35}$ replace the $C(AC)_{P,1-35}$. Positions S and Q of the AC are cleared. The C(SI) are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 49.

LDI — Load Indicators



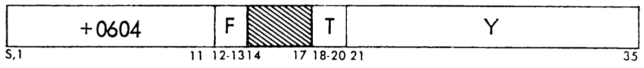
Description. The $C(Y)_{S,1-35}$ replace the $C(SI)_{0-35}$. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 49.

STI — Store Indicators



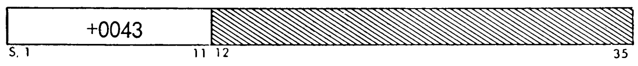
Description. The $C(SI)_{0-35}$ replace the $C(Y)_{S,1-35}$. The C(SI) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 49.

OAI — OR Accumulator to Indicators

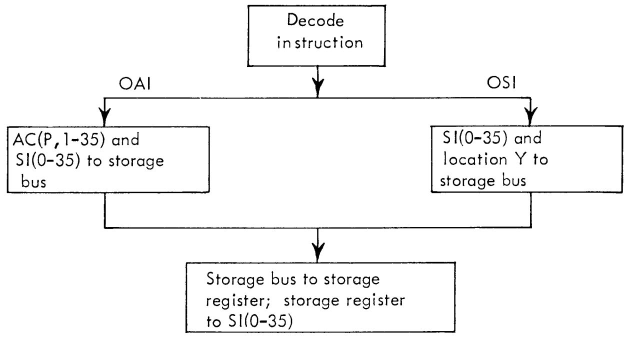


Description. Each bit of the $C(AC)_{P,1-35}$ is matched with the corresponding bit of the $C(SI)_{0-35}$. The C(AC) are unchanged. When the corresponding bit of either (or both) the AC or SI is a 1, a 1 replaces the contents of that position in the SI. When the corresponding bit of both the AC and SI is a 0, a 0 replaces the contents of that position of the SI.

Indicators. None.

Timing: 1 cycle.

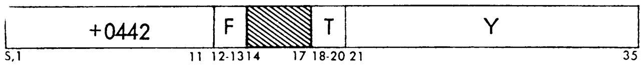
Execution. See Figure 50.



Decode instruction
OAI
OSI
AC(P,1-35) and SI(0-35) to storage bus
SI(0-35) and location Y to storage bus
Storage bus to storage register; storage register to SI(0-35)

Figure 50. OAI and OSI Flow Chart

OSI — OR Storage to Indicators



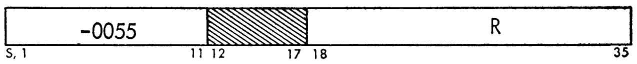
Description. Each bit of the $C(Y)_{S,1-35}$ is matched with the corresponding bit of the $C(SI)_{0-35}$. The C(Y) are unchanged. When the corresponding bit of either location Y or SI (or both) is a 1, a 1 replaces the contents of that position of the SI. When the corresponding bit of both Y and SI is a 0, a 0 replaces the contents of that position in the SI.

Indicators. None.

Timing: 2 cycles

Execution. See Figure 50.

SIL — Set Indicators of Left Half



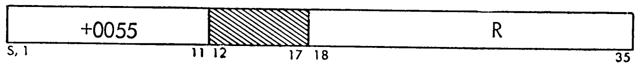
Description. Each bit in positions 18-35 (R) of this instruction is matched with the corresponding bit of the $C(SI)_{0-17}$. The $C(SI)_{18-35}$ and R are unchanged. When the corresponding bit of either (or both) R or the SI is a 1, a 1 replaces the contents of that position in the SI. When the corresponding bit of both the R and the SI is a 0, a 0 replaces the contents of that position in the SI.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 51.

SIR — Set Indicators of Right Half



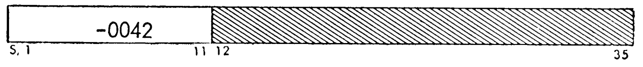
Description. Each bit in positions 18-35 (R) of this instruction is matched with the corresponding bit of the C(SI)$_{18\text{-}35}$. The C(SI)$_{0\text{-}17}$ and R are unchanged. When the corresponding bit of either (or both) R or the SI is a 1, a 1 replaces the contents of that position in the SI. When the corresponding bit of both the R and SI is a 0, a 0 replaces the contents of that position in the SI.

Indicators. None.

Timing: 1 cycle.

Execution. See Figure 52.

RIA — Reset Indicators from Accumulator



Description. Each bit of the C(AC)$_{P,1\text{-}35}$ resets to 0 the corresponding bit of the C(SI)$_{0\text{-}35}$. The C(AC) are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. When the bit in the AC is a 1, a 0 replaces the contents of that position in the SI. When the bit in the AC is a 0, the contents of that position in the SI are unchanged. This is accomplished by taking the contents of the accumulator, bit for bit, and feeding it into a reset input of the sense register. This input will accept only a "1" pulse which turns that position off (0 condition).

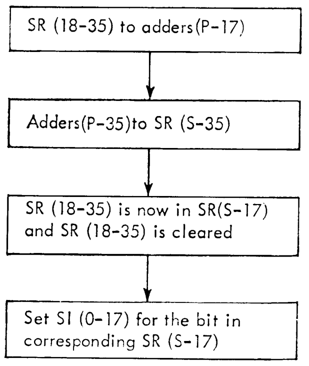

SR (18-35) to adders(P-17)
Adders(P-35)to SR (S-35)
SR (18-35) is now in SR(S-17) and SR (18-35) is cleared
Set SI (0-17) for the bit in corresponding SR (S-17)

Figure 51. SIL Flow Chart

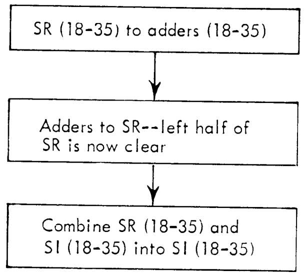
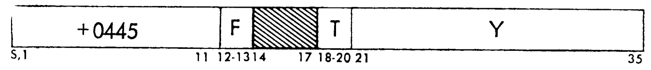

SR (18-35) to adders (18-35)
Adders to SR--left half of SR is now clear
Combine SR (18-35) and SI (18-35) into SI (18-35)

Figure 52. SIR Flow Chart

RIS — Reset Indicators from Storage



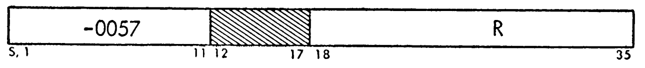
Description. Each bit of the C(Y)$_{S,1\text{-}35}$ resets to 0 the corresponding bit of the C(SI)$_{0\text{-}35}$. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. When the bit in location Y is a 1, a 0 replaces the contents of that position in the SI. When the bit in location Y is a 0, the contents of that position in the SI are unchanged. The operation is identical to that of RIA except that the contents of a storage location, instead of the accumulator, are used to reset the indicators.

RIL — Reset Indicators of Left Half



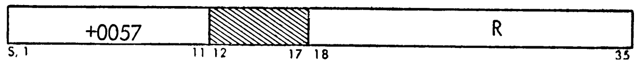
Description. Each bit in positions 18-35 (R) of this instruction resets to 0 the corresponding bit of the C(SI)$_{0\text{-}17}$. The C(SI)$_{18\text{-}35}$ and R are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. When the bit in R is a 1, a 0 replaces the contents of that position in the SI. When the bit in R is a 0, the contents of that position in the SI are unchanged. The operation is identical to that of RIA except that positions 18-35 of the RIL instruction are used to reset positions 0-17 of the sense indicators.

RIR — Reset Indicators of Right Half



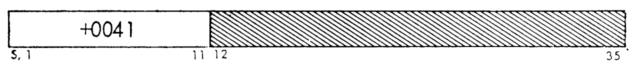
Description. Each bit in positions 18-35 (R) of this instruction resets to 0 the corresponding bit of the C(SI)$_{18\text{-}35}$. The C(SI)$_{0\text{-}17}$ and R are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. When the bit in R is a 1, a 0 replaces the contents of that position in the SI. When the bit in R is a 0, the contents of that position in the SI are unchanged. The operation is identical to that of RIA except that positions 18-35 of the RIR instruction are used to reset positions 18-35 of the sense indicators.

IIA — Invert Indicators from Accumulator



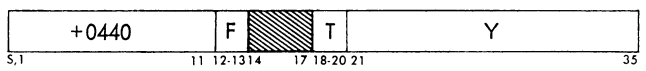
Description. Each bit of the C(AC)$_{P,1-35}$ is matched with the corresponding bit of the C(SI)$_{0-35}$. When the bit in the AC is a 1, the contents of that position in the SI are complemented. When the bit in the AC is a 0, the contents of that position in the SI are unchanged. The C(AC) are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. Sense indicator positions may have "binary input." When this input is used, a "1" pulse fed to the position will reverse its status. For example, if the position holds a 0 and a 1 is fed to it, the position will reverse to a 1 status. Likewise, if the position holds a 1 and a 1 is fed to it, it will flip to a zero status. Zeros fed to the binary input do not affect the position. For the IIA instruction, the C(AC)$_{P,1-35}$ are fed to the binary inputs of the SI(0-35).

IIS — Invert Indicators from Storage



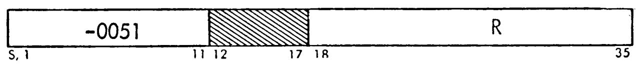
Description. Each bit of the C(Y)$_{S,1-35}$ is matched with the corresponding bit in the C(SI)$_{0-35}$. When the bit in the location Y is a 1, the contents of that position in the SI are complemented. When the bit in location Y is a zero, the contents of that position in the SI are unchanged. The C(Y) are unchanged.

Indicators. None.

Timing: 2 cycles

Execution. The same procedure as IIA except that the contents of storage location Y, instead of the contents of the accumulator, are used to reset the sense indicators.

IIL — Invert Indicators of Left Half



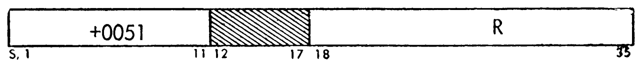
Description. Each bit of positions 18-35 (R) of this instruction is matched with the corresponding bit of the C(SI)$_{0-17}$. The C(SI)$_{18-35}$ and R are unchanged. When the bit in R is a 1, the contents of that position in the SI are complemented. If the bit in R is a zero, the contents of that position in the SI are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. The same as IIA except that 18-35 of this instruction is used in resetting positions 0-17 of the sense indicators.

IIR — Invert Indicators of Right Half



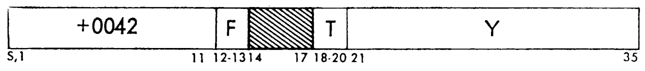
Description. Each bit of positions 18-35 (R) of this instruction is matched with the corresponding bit of the C(SI)$_{18-35}$. The C(SI)$_{0-17}$ and R are unchanged. When the bit in R is a 1, the contents of that position in the SI are complemented. When the bit in R is a zero, the contents of that position in the SI are unchanged.

Indicators. None.

Timing: 1 cycle.

Execution. The same as IIA except that 18-35 of the IIR are used to reset positions 18-35 of the sense indicators.

TIO — Transfer when Indicators On



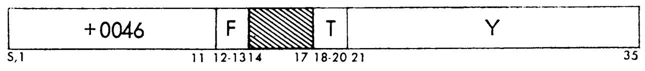
Description. For each bit in the C(AC)$_{P,1-35}$ that is a 1, the corresponding position of the C(SI)$_{0-35}$ is examined. If all the examined positions in the SI contain a 1, the computer takes its next instruction from location Y. If any of the examined positions is not a 1, the computer takes the next sequential instruction. The C(AC) and C(SI) are unchanged.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 53.

TIF — Transfer when Indicators Off



Description. For each bit of the C(AC)$_{P,1-35}$ that is a 1, the corresponding position of the C(SI)$_{0-35}$ is examined. If all examined positions contain 0's, the computer takes its next instruction from location Y. If any of the examined positions does not contain a 0, the computer takes the next sequential instruction. The C(AC) and C(SI) are unchanged.

Indicators. Trap mode.

Timing: 2 cycles

Execution. See Figure 53.

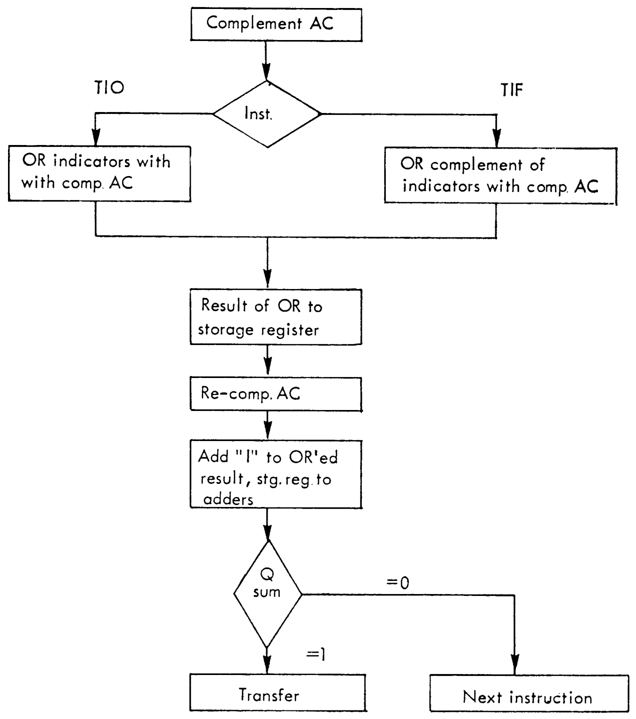
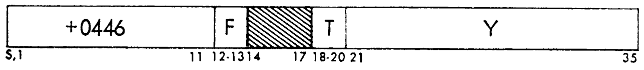

Complement AC
TIO
Inst.
TIF
OR indicators with with comp. AC
OR complement of indicators with comp. AC
Result of OR to storage register
Re-comp. AC
Add "1" to OR'ed result, stg. reg. to adders
Q sum
=0
=1
Transfer
Next instruction

Figure 53. TIO and TIF Flow Chart

ONT — On Test for Indicators



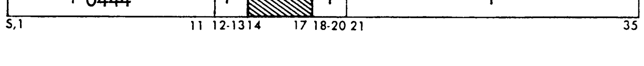
Description. For each bit in the $C(Y)_{S,1\text{-}35}$ that is a 1, the corresponding position of the $C(SI)_{0\text{-}35}$ is examined. If all the examined positions in the SI contain a 1, the computer skips the next instruction and proceeds from there. If any of the examined positions do not contain 1's, the computer takes the next sequential instruction. The C(Y) and C(SI) are unchanged.

Indicators. None.

Timing: 4 cycles

Execution. See Figure 54.

OFT — Off Test for Indicators



Description. For each bit of the $C(Y)_{S,1\text{-}35}$ that is a 1, the corresponding position of the $C(SI)_{0\text{-}35}$ is examined. If all the examined positions contain 0's, the computer skips the next instruction and proceeds from there. If any of the examined positions does not contain a 0, the computer takes the next sequential instruction. The C(Y) and C(SI) are unchanged.

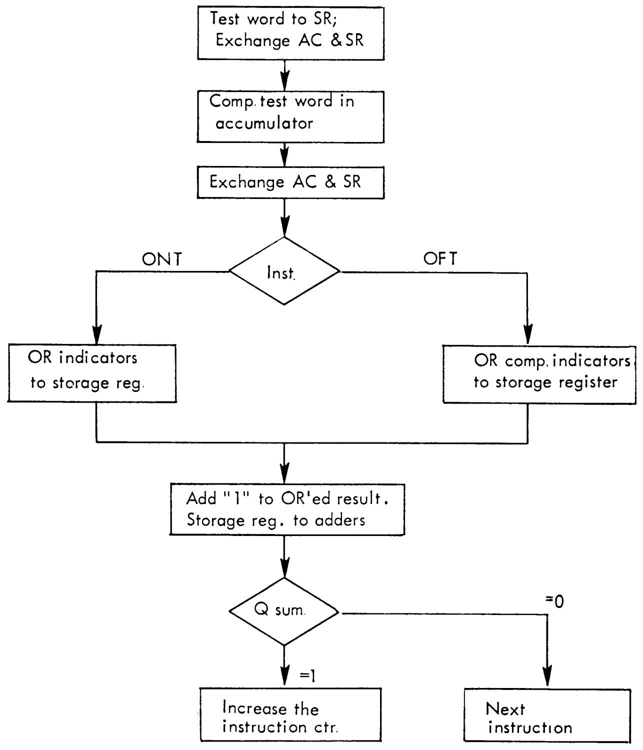

Test word to SR; Exchange AC & SR
Comp. test word in accumulator
Exchange AC & SR
ONT
Inst.
OFT
OR indicators to storage reg.
OR comp. indicators to storage register
Add "1" to OR'ed result. Storage reg. to adders
Q sum
=0
=1
Increase the instruction ctr.
Next instruction

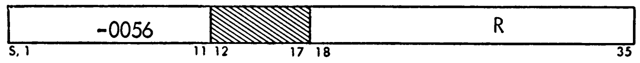
Figure 54. ONT and OFT Flow Chart

Indicators. None.

Timing: 4 cycles

Execution. See Figure 54.

LNT — Left Half Indicators on Test

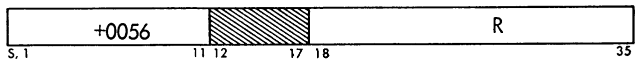


Description. For each bit in positions 18-35 (R) of this instruction that is a one, the corresponding position of the $C(SI)_{0\text{-}17}$ is examined. The C(SI) and R are unchanged. If all the examined positions contain a 1, the computer skips the next instruction and proceeds from there. If any of the examined positions does not contain a 1, the computer takes the next sequential instruction.

Indicators. None.

Timing: 3 cycles

RNT — Right Half Indicators on Test

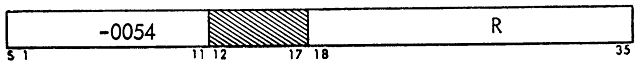


Description. For each bit in positions 18-35 (R) of this instruction that is a 1, the corresponding position of the C(SI)$_{18\text{-}35}$ is examined. The C(SI) and R are unchanged. If all the examined positions contain a 1, the computer skips the next instruction and proceeds from there. If any of the examined positions does not contain a 1, the computer takes the next sequential instruction.

Indicators. None.

Timing: 3 cycles

LFT — Left Half Indicators Off Test

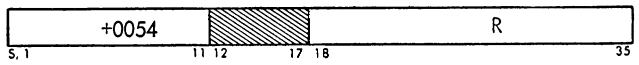


Description. For each bit in positions 18-35 (R) of this instruction that is a 1, the corresponding position of the C(SI)$_{0\text{-}17}$ is examined. The C(SI) and R are unchanged. If all the examined positions contain a 0, the computer skips the next instruction and proceeds from there. If any of the examined positions does not contain a zero, the computer takes the next sequential instruction.

Indicators. None.

Timing: 3 cycles

RFT — Right Half Indicators Off Test



Description. For each bit in positions 18-35 (R) of this instruction that is a one, the corresponding position of C(SI)$_{18\text{-}35}$ is examined. If all the examined positions contain a zero, the computer skips the next instruction and proceeds from there. If any of the examined positions does not contain a zero, the computer takes the next sequential instruction. The C(SI) and R are unchanged.

Indicators. None.

Timing: 3 cycles

Convert Instructions

The convert operations enable a program to have very rapid access to information stored in tables in core storage. A single convert instruction can perform a series of table look-up operations by making multiple references to core storage. Three such convert operations are available in the computer.

To illustrate the method of execution of these convert instructions, the following section describes the CONVERT BY REPLACEMENT FROM THE MQ (CRQ) instruction. The following two definitions will apply throughout this description:

Argument—The known reference factor necessary to find a desired item in a table.

Function—The unknown factor in a table associated with a known reference factor (argument).

The contents of the MQ are interpreted as six 6-bit quantities. Each of these quantities may be considered as a 6-bit binary integer and will be designated as L1, L2, ..., L6 (Figure 55).

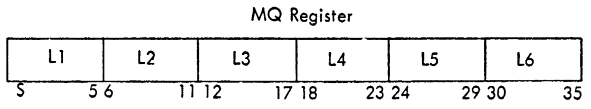
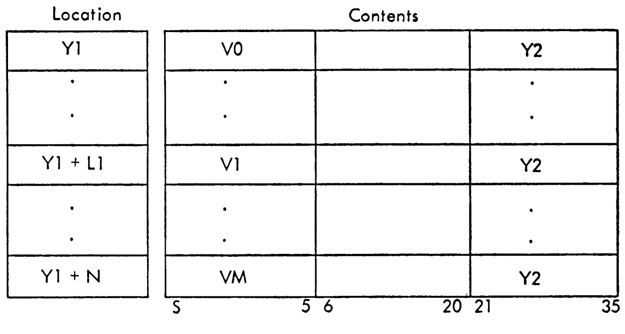


Figure 55. MQ Register

The number, Y1, contained in the address part of the CRQ instruction is interpreted by the instruction as the *address* of the first location (origin) of a table in core storage. The format of a typical table is shown in Figure 56. The 6-bit binary integer, L1, is taken as the first argument and the address Y1 + L1 is formed. The instruction then looks up the word located at Y1 + L1. The left-most six bits of this word, positions S, 1-5, are taken as the desired function V1. This 6-bit number, V1, then replaces the number L1 in the MQ. The right-most 15 bits of this word, positions 21-35, are interpreted as an address, Y2, specifying the origin of a second table in core storage. The number L2 is then taken as the second argument and a reference to the second table is made at location Y2 + L2. The contents of Y2 + L2, positions S, 1-5, make up the second function, V2, and replace L2 in the MQ. Positions 21-35 of this word are interpreted as a third address, Y3, the origin of a third table in core storage, and the process continues.

The function V1 replaces L1 in the MQ through a shifting operation. The MQ is shifted left six positions and the bits of L1 are shifted out of position S of the MQ and are lost. The number V1 then replaces the contents of the MQ, positions 30-35 (Figure 57).



Location	Contents		
Y1	V0		Y2
. .	. .		. .
Y1 + L1	V1		Y2
. .	. .		. .
Y1 + N	VM		Y2
	S 5	6 20	21 35

Figure 56. Convert Table

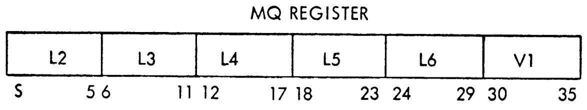


Figure 57. MQ Register

Thus, the CRQ instruction provides for replacement of the contents of the MQ from left to right.

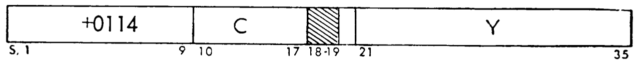
The number of such table references made by a single convert instruction is specified by the count field, positions 10-17, of the instruction. If a count of six is given, six numbers V1, V2, ..., V6 will occupy the exact MQ positions originally containing the six corresponding arguments L1, L2, ..., L6. If a count of one was specified for a CRQ instruction, the final contents of the MQ would be as shown in Figure 57. When a count of more than six is specified, the values taken from the tables during the first six references will be used for additional table references. For example, V1 = L7, V2 = L8, and so on.

After the last function, Vn, has been placed in the MQ, the location in core storage from which Vn has been taken contains as its address part a number, Yn. If the tag field of the convert instruction contains a one, the number Yn replaces the contents of index register 1. This provides a convenient method for the program to determine where the last storage table reference has been made or where the next table reference is to be made. (Only index register 1 can be used.)

The CONVERT BY REPLACEMENT FROM THE AC (CVR) instruction is analogous to the CRQ instruction. For this instruction the $C(AC)_{P,1\text{-}35}$ rather than the C(MQ) are interpreted as the 6-bit numbers L1, L2, ..., L6. Also, the six-place shifts which occur during the execution of the instruction are right shifts rather than left shifts. Thus, for the CVR instruction, the replacement takes place right to left, whereas, for the CRQ, the replacement takes place from left to right.

The CAQ instruction interprets the C (MQ) as six 6-bit quantities L1, L2, ..., L6 and uses these numbers as arguments in the same manner as the CRQ instruction. However, instead of replacing the numbers L1, L2, ..., L6, the contents of the looked-up words are added into the AC. The address parts of these words are then used as origins for additional look-ups in the usual manner. Addition into the AC is logical, with the S position of the word being added into the P position of the AC. After an argument has been used for a table reference, the C (MQ) are rotated left so that bits leaving position S enter position 35 of the MQ. Thus, if a count of six is specified for a CAQ instruction, the final and original C (MQ) are identical.

CVR — Convert by Replacement from the AC



Description. This instruction treats the $C(AC)_{P,1\text{-}35}$ as six 6-bit quantities and replaces the first C of these quantities by values from tables in core storage. Position S of the AC is unchanged. NOTE: Bit position Q is not cleared and will OR with the first function, if present initially.

Indicators. None.

Timing: 2-8 cycles, modification 6

Execution. The instruction is executed in the following steps:

1. The address part (Y) replaces the $C(SR)_{21\text{-}35}$.

2. The count field (C) is placed in the shift register.

3. The contents of the shift register are tested. If the register contains zero, step 4a follows. If the register is non-zero, step 4b follows.

4a. If position 20 of this instruction contains a 1, the $C(SR)_{21\text{-}35}$ replace the contents of index register 1 (XRA), and the computer takes the next sequential instruction. If position 20 contains a 0, the computer proceeds directly to the next sequential instruction.

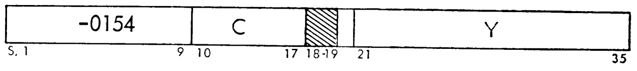
4b. The $C(SR)_{21\text{-}35}$ are added to the $C(AC)_{30\text{-}35}$ to form an address (X). The C (X) replace the C (SR).

5. The $C(AC)_{Q,P,1\text{-}35}$ are shifted right six places. Positions vacated are filled with zeros.

6. The $C(SR)_{S,1\text{-}5}$ replace the $C(AC)_{P,1\text{-}5}$. However, if position Q of the AC initially contains a 1, it will be shifted to position 5 of the AC during step 5. This 1 in position 5 of the AC will remain regardless of the contents of position 5 of the SR.

7. The contents of the shift register are decreased by one and the computer returns to step 3.

CRQ — Convert by Replacement from the MQ



Description. This instruction treats the C (MQ) as six 6-bit quantities and replaces the first C of these quantities by values from tables in core storage.

Indicators. None.

Timing: 2-8 cycles, modification 6

Execution. The instruction is executed in the following steps:

1. The address part (Y) replaces the $C(SR)_{21\text{-}35}$.

2. The count field (C) is placed in the shift register.

3. The contents of the shift register are tested. If the register contains 0, step 4a follows. If the register is not 0, step 4b follows.

4a. If position 20 of this instruction contains a 1, the c(SR)$_{21-35}$ replace the contents of XRA and the computer proceeds to the next sequential instruction. If position 20 contains a 0, the computer proceeds directly to the next sequential instruction.

4b. The c(SR)$_{21-35}$ are added to the c(MQ)$_{S,1-5}$ to form an address (X). The c(x) then replace the c(SR).

5. The c(MQ) are shifted left six places. Bits shifted out of position S of the MQ are lost. Positions vacated are filled with zeros.

6. The c(SR)$_{S,1-5}$ replace the c(MQ)$_{30-35}$.

7. The contents of the shift register are decreased by one, and the computer returns to step 3.

CAQ — Convert by Addition from the MQ

-0114	C	▨	Y
S, 1 … 9	10 … 17	18-19	21 … 35

Description. This instruction treats the c(MQ) as six 6-bit quantities. The first C of these quantities are used in making references to tables in core storage. Words selected by the references are added to the c(AC)$_{Q,P,1-35}$. Position S of the AC is unchanged. NOTE: Care should be taken that the binary sum of the quantities added from 21-35 does not carry into position 19 of the AC.

Indicators. None.

Timing: 2-8 cycles, modification 6

Execution. The instruction is executed in the following steps:

1. The address part (Y) replaces the c(SR)$_{21-35}$.

2. The count field (C) is placed in the shift register.

3. The contents of the shift register are tested. If the register contains 0, step 4a follows. If the register is not 0, step 4b follows.

4a. If position 20 of this instruction contains a 1, the c(SR)$_{21-35}$ replace the contents of XRA, and the computer proceeds to the next sequential instruction. If position 20 contains a 0, the computer proceeds directly to the next sequential instruction.

4b. The c(SR)$_{21-35}$ are added to the c(MQ)$_{S,1-5}$ to form an address (X). The c(x) then replace the c(SR).

5. The c(MQ) are rotated six positions to the left. Bits leaving position S of the MQ enter position 35.

6. The c(SR)$_{S,1-35}$ are added to the c(AC)$_{Q,P,1-35}$ and the sum replaces the c(AC)$_{Q,P,1-35}$. The sign position of the SR is added into position P of the AC, and the sign of the AC is disregarded. NOTE: Even though AC overflow is possible, the overflow indicator is not affected by this instruction.

7. The contents of the shift register are reduced by one and the computer returns to step 3.

The reader is referred to the CONVERT programming examples contained in the programming section of this manual for possible uses of these instructions.

Input-Output Operations

As has been previously stated, the address part of an instruction may refer either to a location in core storage, the length of shift, or may be interpreted as a part of the operation code itself.

The identifying number for the various input-output units appears in the address part of the instruction. For tapes, card machines and printers the address part specifies both the particular I-O unit and the data channel to which it is attached. Channels A through H are specified by the numbers 1 through 10, respectively, appearing as the first two digits of an octal five-digit address. The last three digits specify the I-O unit. If the I-O unit is a tape, the mode of operation, either binary or BCD, is also specified by the address.

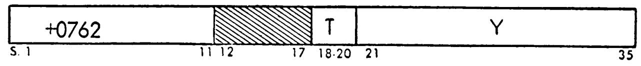
The addresses of the input-output devices are shown below:

DEVICE	CHANNEL	BCD ADDRESS	BINARY ADDRESS
Tapes	A	1201-1212	1221-1232
	B	2201-2212	2221-2232
	C	3201-3212	3221-3232
	D	4201-4212	4221-4232
	E	5201-5212	5221-5232
	F	6201-6212	6221-6232
	G	7201-7212	7221-7232
	H	10201-10212	10221-10232
Card Reader	A	1321	
	B	2321	
	C	3321	
	D	4321	
	E	5321	
	F	6321	
	G	7321	
	H	10321	
Card Punch	A	1341	
	B	2341	
	C	3341	
	D	4341	
	E	5341	
	F	6341	
	G	7341	
	H	10341	
		NORMAL	BINARY
Printer	A	1361	1362
	B	2361	2362
	C	3361	3362
	D	4361	4362
	E	5361	5362
	F	6361	6362
	G	7361	7362
	H	10361	10362

In the following instruction description, only the operation code will be shown. The address part will appear in the normal code of Y. All input-output instructions may be trapped using the compatibility feature.

Since it is possible to create magnetic tapes of mixed density, precautions should be taken so that all read tape instructions are executed when the tape unit is set to the density in which the tape was recorded.

RDS — Read Select



Description. This instruction causes the computer to prepare to read information, from the I-O device specified by Y, into core storage. If Y specifies a tape, printer, or card reader, Y also specifies the channel to which the device is attached.

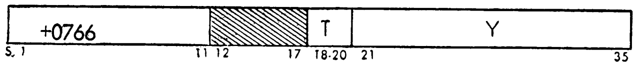
Any attempt to read in one density a tape that was recorded in the other density will result in both detected and undetected errors. When RDS is used to skip tape, proper tape density must also be used.

Indicators. Simulate, end of file. (See device being used.)

Timing: 2 cycles, modification 8

Execution. When a channel is designated, a RESET AND LOAD CHANNEL instruction must be given within the specified time following the RDS, or the I-O device will be logically disconnected from the computer and one record will be passed. (See each device for timings.) No other select instruction should be inserted between the RDS and its associated RCH. When an RDS specifies channel operation, only positions 28-35 of the address part of the instruction are subject to effective address modification.

WRS — Write Select



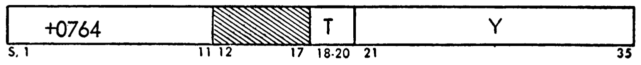
Description. This instruction causes the computer to prepare to write information from storage to the I-O device specified by Y. If Y specifies a tape, printer or card punch, Y also specifies the channel to which the device is attached.

Indicators. Simulate, end of tape. (See device being used.)

Timing: 2 cycles, modification 8

Execution. When a channel is designated, a RESET AND LOAD CHANNEL instruction must be given within specified time following the WRS, or the I-O device will be logically disconnected from the computer and, if the WRS specified a tape, a blank section of tape will be written. No other select instruction should be inserted between the WRS and its associated RCH. If the end of tape reflective spot is encountered during the execution of a WRS, the end-of-tape indicator in the proper channel will be turned on. When a WRS specifies an I-O device attached to a channel, only positions 28-35 of the address part of the instruction are subject to effective address modification.

BSR — Backspace Record



Description. This instruction causes the tape designated by Y to move backward until recorded information is reached and then to continue this backward motion over information until an end-of-record gap or load point is encountered.

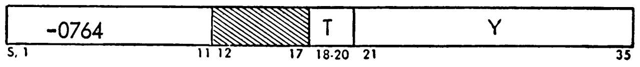
The tape unit must be in the proper density mode (the same as the tape being backspaced) before the BSR is executed, or tape errors will occur.

Indicators. Beginning of tape and simulate.

Timing: 2 cycles, modification 8

Execution. If the tape designated by Y is positioned at its load point, a BSR is interpreted as a no-operation and the beginning-of-tape indicator in the proper channel is turned on. If load point is encountered before information is encountered, the BOT indicator is turned on. Only positions 28-35 of the address part of this instruction are subject to effective address modification.

BSF — Backspace File



Description. This instruction causes the tape designated by Y to move backward until recorded information is reached and then to continue this backward motion over information and end-of-record gaps until an end-of-file record or the load point is encountered.

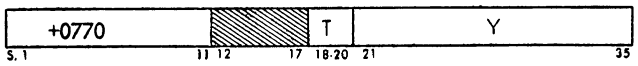
The tape unit must be in the proper density mode (the same as the tape being backspaced) before the BSF is executed or tape errors will occur.

Indicators. Beginning of tape, simulate.

Timing: 2 cycles, modification 8

Execution. If a BSF is given to a tape positioned at its load point, the BSF is interpreted as a no-operation and the beginning-of-tape indicator in the proper channel is turned on. If load point is encountered before an end-of-file record, the beginning-of-tape indicator is turned on. Only positions 28-35 of the address part of this instruction are subject to effective address modification.

WEF — Write End-of-File



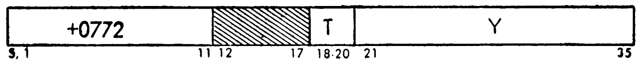
Description. This instruction causes the tape designated by Y to write an end-of-file gap followed by a tape mark (and its check character) on the tape.

Indicators. End of tape, simulate.

Timing: 2 cycles, modification 8

Execution. If an end of tape reflective spot is passed over during the execution of a WEF, the end-of-tape indicator in the proper channel is turned on. Only positions 28-35 of the address part of this instruction are subject to effective address modification.

REW — Rewind



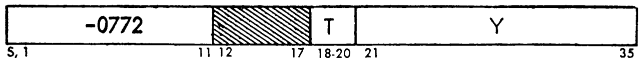
Description. This instruction causes the tape designated by Y to rewind its tape to the load point position.

Indicators. Simulate.

Timing: 2 cycles, modification 8

Execution. If the tape is positioned at its load point at the time the REW is interpreted, the instruction is treated as a no-operation. Only positions 28-35 of the address part of this instruction are subject to effective address modification.

RUN — Rewind and Unload



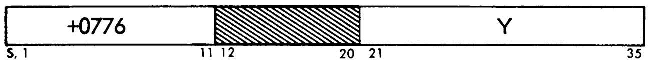
Description. The tape unit designated by the address portion of this instruction will be rewound and then put into an automatic unload status.

Indicators. None.

Timing. 2 cycles, modification 8.

Execution. This instruction will be executed in the same manner as a rewind instruction except that it will never be treated as a no-operation if the tape is positioned at load point when execution occurs. After the rewind, a normal unload operation will occur as if the operator had depressed the unload key. Only positions 28-35 of this instruction are subject to modification by index register action. With a tape at load point, this instruction will cause an unload operation only.

SDN — Set Density



Description. The address portion of this instruction will determine which density mode is being used for a given tape operation.

Indicators. None.

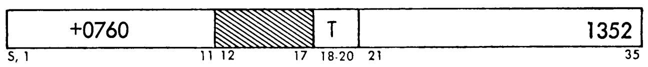
Timing. 2 cycles.

Execution. The address portion of this instruction will include the data channel being used, tape unit number, and the density mode as follows:

CHANNEL	HIGH DENSITY	LOW DENSITY
A	1221- 1232	1201- 1212
B	2221- 2232	2201- 2212
C	3221- 3232	3201- 3212
D	4221- 4232	4201- 4212
E	5221- 5232	5201- 5212
F	6221- 6232	6201- 6212
G	7221- 7232	7201- 7212
H	10221-10232	10201-10212

Care must be taken in using this instruction because of the possibility of changing the density mode in the middle of a tape. When power is first applied to the 7090 system, all tape units will automatically be placed in high density status.

RDCA — Reset Data Channel A



Description: This instruction resets all registers and indicators in the designated data channel. All transmission is terminated and tape or other selected units are immediately disconnected. If the instruction is executed while a tape is in motion, the tape is stopped *immediately* regardless of the position of the tape head with regard to the inter-record gap. Any instructions which have been stacked will be lost and status indicators previously set by an enable instruction will be turned off.

Timing: 2 cycles.

Execution: All registers, indicators and control elements that are turned off by the reset key on the data channel console are turned off by this instruction. Since the primary purpose of this instruction is to clear a channel after it has been "hung up" by selecting a unit that is not ready, no attempt is made to synchronize its operation with I-O units. *Use of the instruction for other purposes may result in unpredictable operation.* Address modification may cause the operation code to be changed. Instruction codes and mnemonics for each channel are:

RDCB	+0760...2352
RDCC	+0760...3352
RDCD	+0760...4352
RDCE	+0760...5352
RDCF	+0760...6352
RDCG	+0760...7352
RDCH	+0760..10352

The following program is an example of using the RDC to determine ready status of tape units attached to channel A. The program stores a bit pattern in location STAT such that a 1-bit signifies the tape unit is ready (position 35 corresponds to unit 1, 34 to unit 2, etc.).

```
LOCATION  INSTRUCTION  ADDRESS        COMMENTS
          TCOA         *
          STZ          STAT
          AXT          10, 1
          CAL          ONE
READY     BSRA         11, 1
          CRQ          0, ,15         Delay is approximately
                                        30 microseconds
          TCOA         *+2
          ORS          STAT           Ready
          ALS          1              Not Ready
          RDCA
          CRQ          0, ,15         Delay approximately
                                        30 microseconds
          TIX          READY, 1, 1
ONE       HTR          1
STAT      OCT          0
```

In this program, the RDC instruction serves two purposes:

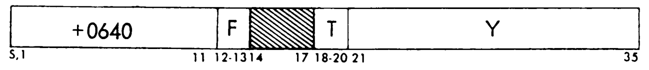
1. If the tape is not ready, the channel is cleared so that the next I-O instruction will be accepted (in the CPU).

2. If the tape is ready, execution of the RDC will effectively stop the backspace operation before it gets started so that the tape does not move from its original position.

Input-Output Transmission Operations

Instructions that either send commands to a data channel or store information from data channel registers are classified as input-output transmission operations. These instructions are described in groups of eight. All eight instructions within a group are identical in function and differ only in that each refers specifically to one of the eight possible data channels. The instructions will be described for data channel A.

SCHA — Store Channel A



Description. This instruction replaces the C(Y) with the contents of the channel A address, location, and operation registers. If channel A is not attached to the computer when the SCHA is given, the C(Y) are cleared by the SCHA.

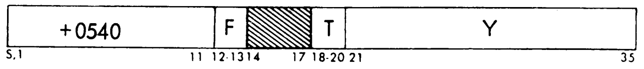
Indicators. None.

Timing: 2 cycles

Execution. The $C(Y)_{21-35}$ are replaced by the C(AR), the $C(Y)_{3-17}$ are replaced by the C(LR), and the $C(Y)_{S,1,2,19}$ are replaced by the contents of the operation register. An SCHA instruction may be executed at any time, regardless of whether or not the specified channel is in operation. If the channel is in operation and the channel registers are in the process of being changed, the execution of the SCHA will be delayed until the change has been completed. Note that the C(AR) will be one greater than the storage location of the last word involved in data transmission and that the C(LR) are one greater than the storage location from which the current command was taken. A channel relinquishes priority between the time the last word is transmitted by an IOCP or IOSP command and the arrival of a subsequent channel command. Therefore, an SCH may store an address which is one greater than the address of the last word transmitted by the IOCP or IOSP commands.

INSTRUCTION	CODE	NAME
SCHB	−0640	Store Channel B
SCHC	+0641	Store Channel C
SCHD	−0641	Store Channel D
SCHE	+0642	Store Channel E
SCHF	−0642	Store Channel F
SCHG	+0643	Store Channel G
SCHH	−0643	Store Channel H

RCHA — Reset and Load Channel A



Description. If channel A has been selected by either an RDS or WRS, the $C(Y)_{S,1,2,19}$ replace the channel operation register, $C(Y)_{3-17}$ replace the C(WR) and the

C(Y)$_{21-35}$ replace the C(AR). In addition, the number Y plus one replaces the C(LR).

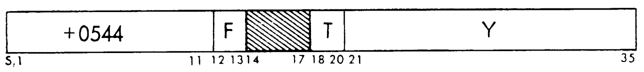
Indicators. I-O Check.

Timing: 3 cycles

Execution. If channel A is not selected when the RCHA is given, the RCHA executes normally but the I-O indicator is turned on (If the command loaded by the RCHA specifies indirect addressing, it will not occur). For each RDS or WRS, the corresponding RCHA must be given if any transmission between storage and the selected I-O device is to take place. If a second RCHA is given at a later time, the order is executed immediately. No other select instruction should be inserted between the RCH and its associated WRS or RDS. See "Programmed Interruption of a Data Channel" for further details regarding this type of operation.

INSTRUCTION	CODE	NAME
RCHB	−0540	Reset and Load Channel B
RCHC	+0541	Reset and Load Channel C
RCHD	−0541	Reset and Load Channel D
RCHE	+0542	Reset and Load Channel E
RCHF	−0542	Reset and Load Channel F
RCHG	+0543	Reset and Load Channel G
RCHH	−0543	Reset and Load Channel H

LCHA — Load Channel A



Description. If the data channel has been selected, the computer delays until an IOCT, IORT, or IOST command is processed for channel A or the channel leaves operation. After an IOCT, IORT, or IOST command has been executed by the channel A, the LCHA is executed as shown below.

Indicators. I-O check.

Timing: 3 cycles, modification 8

Execution. The C(Y)$_{S,1,2,19}$ replace the contents of channel A operation register. C(Y)$_{3-17}$ replace the C(WR), the C(Y)$_{21-35}$ replace the C(AR), and the number Y plus one replaces the C(LR). If an LCHA is issued and either (1) the channel is not selected, or (2) channel A is selected but an IOCT, IORT, or IOST command is not executed before the channel disconnects, the I-O check indicator is turned on and the LCHA is treated as a no-operation.

INSTRUCTION	CODE	NAME
LCHB	−0544	Load Channel B
LCHC	+0545	Load Channel C
LCHD	−0545	Load Channel D
LCHE	+0546	Load Channel E
LCHF	−0546	Load Channel F
LCHG	+0547	Load Channel G
LCHH	−0547	Load Channel H

Data Channel Commands

The eight types of data channel commands are described in much the same manner as other computer instructions. The following steps list command conditions:

1. The letter Y in the address part (21-35) of a command is used to denote a core storage location.

2. The letter C in the decrement part (3-17) of a command is used to denote a word count amount.

3. The numerical operation code is shown by an octal digit in the prefix part (S, 1 and 2). The digit may be visually converted to its binary equivalent for reference to the bit pattern actually used.

4. Indirect addressing of data channel commands is possible on the 7090 system. Position 18 of the command contains the flag bit. Thus, with a command having an address part (Y) and a one in position 18, the address part of location Y replaces the address part of the command before it is executed. With a zero word count, indirect addressing does not occur on IOCP or IOSP commands. Indirect addressing will occur only on read or write operations.

5. Separate commands have not been formulated to handle bit position 19. Instead, a fifth character (N—denoting non-transmit) is appended to the mnemonic codes used for positions S, 1, and 2. This type of command is used for read operations only.

6. Seven of the eight commands deal with data transmission. The codes for these commands all contain the letters "I-O".

7. The eighth command is a transfer in channel command.

8. The word disconnected means that the units involved are separated logically rather than physically. When a disconnect is signalled, it may be delayed depending on the operation being performed. For example:

a. Write Tape. Disconnect will not occur until the end of record gap is written. This permits a programmed ETT or TRC test to be valid if given after the channel leaves operation (disconnects).

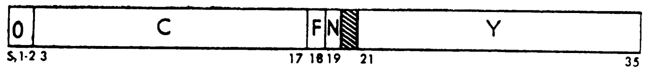
b. Read Tape. Disconnect will not occur until tape control circuits and tape units have accepted the read instruction. This assures that any read select instruction will move tape before the disconnect occurs even if an immediate disconnect is programmed. Disconnect will not occur between the last word of the record and the interrogation of the longitudinal redundancy check character (LRC). This assures that with a channel programmed to read an entire record, a TRC test, given after the channel leaves operation, will have tested the LRC character.

Recognition of an end of file, while reading tape or cards, causes a disconnect regardless of the command being used. The tape unit does not leave operation until the LRC character has been checked and the end of record reached. An IORP, IORT, IOSP, or IOST command will not recognize a logical end of record on an end of file.

c. *Card Machines.* Unless a disconnect is programmed immediately following a transmission or end of record time, the disconnect will generally be delayed until the next transmission or end of record time. For example, if the channel is loaded with an IOCD command (with zero word count) ten milliseconds after end of record time, an extra machine cycle will occur. The disconnect then occurs at the 9-left transmission time of this extra cycle.

IN EXECUTION descriptions of the I-O commands, tape is assumed to be the I-O device. However, mechanical motion on the tape (from record to record) may be compared to card equipment (from card to card or line to line) on the printer. The descriptions may then be applied to I-O devices other than tape.

IOCD — Input-Output under Count Control and Disconnect



Description. C words are transmitted between an I-O device and core storage beginning with location Y. The data transmission is under control of the count (C) field only.

Indicators. See I-O device being used.

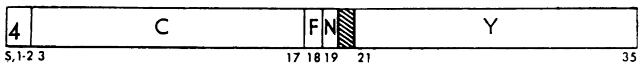
Timing: See I-O device being used.

Execution

Read Operation. If the word count has been reduced to zero before a record gap is reached, the channel leaves operation and tape motion will continue until a record gap is reached. No transmission will occur during this time. If a record gap is encountered when the word count is not zero, the gap is ignored and reading continues from the next record. An IOCD command with a 0 word count, loaded at the end of record will disconnect the channel.

Write Operation. When C words have been written on tape, a record gap is written. If this command is given at the beginning of a record and C is initially zero, approximately 3¾ inches of blank tape will be written before the record gap is written. The channel is then disconnected.

IOCP — Input-Output under Count Control and Proceed



Description. C words are transmitted between an I-O device and core storage location Y. The data transmission is under control of the count field only. When C is reduced to zero, or is initially zero, the next sequential command is brought into the data channel and executed.

Indicators. See I-O device being used.

Timing. See I-O device being used.

Execution

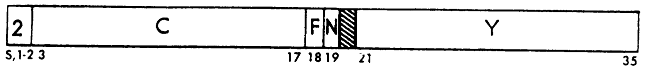
Read Operation. C words are read from tape and stored in consecutive storage locations beginning with location Y. When the word count has been reduced to zero, the channel takes its next command in sequence and executes it.

If the word count is reduced to zero by the last word of a record and the next command is an IORP, IOSP, IORT, or IOST, the present end of record will be recognized. Unlike the IOSP, the IOCP command need not proceed within 11 cycles in order to recognize the present end of record.

Write Operation. C words from storage beginning with location Y are written on tape. When the specified words have been written, the channel proceeds to the next sequential command. An end of record is not written on tape when the word count is reduced to zero.

In printing or the punching of cards, if the word count is reduced to zero on the 12-row right transmission point, and the next command is an IORP or IORT, the end of the present machine cycle (record) will be recognized.

IORP — Input-Output of a Record and Proceed



Description. See "Execution."

Indicators. See I-O device being used.

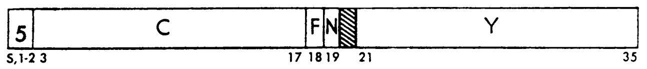
Timing: See I-O device being used.

Execution

Read Operation. Words are transmitted from tape and stored in consecutive storage locations until either an end of record is encountered or the word count has been reduced to zero. If the word count is reduced to zero (or is initially zero) before an end of record is reached, the rest of the words in that record are skipped without transmission to storage. When the record gap is reached, the channel takes the next sequential command.

Write Operation. When C words have been written, a record gap is written and the channel proceeds to the next sequential command.

IOCT — Input-Output under Count Control and Transfer



Description. C words are transmitted between an I-O device and storage beginning with location Y. The data transmission is under control of the count field only. When C is reduced to zero, the next command is taken from a core location specified by the load channel instruction in the main program or disconnects (and traps if the channel is enabled) if no load channel instruction is waiting. If this command is loaded by an RCH or LCH instruction and C is initially zero, the channel is immediately disconnected unless restricted by the operation.

Indicators. Command Trap.

Timing: See I-O device being used.

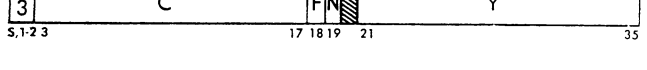
Execution

Read Operation. Execution of the command is under control of the count field only and end of record gaps are ignored. If the word count is reduced to zero by the last word of a record and the next command is an IORP, IOSP, IORT, or IOST, the present end of record will be recognized. Unlike the IOST command, the LCH instruction need not load the channel within 11 cycles in order to recognize the present end of record.

Write Operation. An end of record will not be written after C words have been written if the LCH instruction results in bringing into the channel a new word count.

In printing or the punching of cards, if the word count is reduced to zero on the 12-row right transmission point and the next command is an IORP or IORT, the end of the present machine cycle (record) will be recognized.

IORT — Input-Output of a Record and Transfer



Description. See "Execution."

Indicators. Command trap.

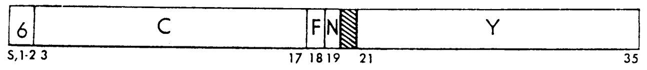
Timing: See I-O device being used.

Execution

Read Operation. Words are transmitted from tape and stored in consecutive storage locations until either an end of record is encountered or the word count has been reduced to zero. If the word count is reduced to zero (or is initially zero) before an end of record is reached, the rest of the words in that record are skipped without transmission to storage. When the record gap is reached, the next command is taken from a core location specified by the LCH instruction in the main program or disconnects (and traps if the channel is enabled) if no LCH is waiting.

Write Operation. When C words have been written, a record gap is written and the channel takes its next command from a core location specified by the LCH instruction in the main program or disconnects (and traps if the channel is enabled) if no LCH instruction is waiting.

IOSP — Input-Output until Signal, then Proceed



Description. See "Execution."

Indicators. See I-O device being used.

Timing: See I-O device being used.

Execution

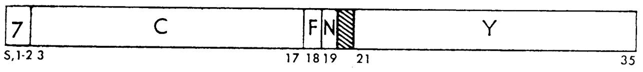
Read Operation. Words are read into consecutive storage locations beginning with location Y until either the contents of the word counter are reduced to zero or an end of record is reached. When either event occurs, the channel proceeds immediately to the next sequential command and executes it.

With a tape read operation and an IOSP or IOST command whose word count is reduced to zero by the last word in the record and whose next command is an IORP, IORT, IOSP, or IOST, this next command will normally enter the channel in time to recognize the present end of record. This next command transmits no data and is effectively skipped. If this next command cannot enter the channel within 11 machine cycles, the IORP, IORT, IOSP, or IOST command will not recognize the present end of record gap, but will process the following record. To determine if this sequence can be safely programmed, three cycles should be allowed if the CPU is processing a TCO instruction (five cycles if other instructions are being processed) plus one cycle for each channel in operation, plus one cycle for each channel programmed with proceed commands, plus one cycle for each channel which may process a TCH at this time, plus one cycle for each channel which may have indirect addressing at this time. If the total exceeds 11 cycles, the sequence described must not be used.

Write Operation. C words are written on tape beginning with storage location Y. When the specified words have been written, the channel proceeds to the next sequential command. An end of record is not written on tape when the word count is reduced to zero. Note that this is identical to the operation of the IOCP.

In printing or the punching of cards, if the word count is reduced to zero on the 12-row right transmission point and the next command is an IORP or IORT, the end of the present machine cycle (record) will be recognized.

IOST — Input-Output until Signal then Transfer



Description. See "Execution."

Indicators. See I-O device being used.

Timing: See I-O device being used.

Execution

Read Operation. Words are read into consecutive storage locations beginning with location Y until either the contents of the word counter are reduced to zero or an end of record is reached. When either event occurs, the channel takes its next command from a core location specified by the LCH instruction in the main program or disconnects (and traps if the channel is enabled) if no LCH instruction is waiting.

With a tape read operation and an IOST command whose word count is reduced to zero by the last word in the record and whose next command is an IORP, IORT, IOSP, or IOST, this next command will normally enter the channel in time to recognize the present end of record. This next command transmits no data and is effectively skipped. If this next command cannot enter the channel within 11 cycles, the IORP, IORT, IOSP, or IOST command will not recognize the present end of record gap but will process the following record. To determine if this sequence can be safely programmed, two cycles should be allowed for the LCH instruction plus one cycle for each channel in operation, plus one cycle for each channel programmed with proceed commands, plus one cycle for each channel using indirect addressing. If the total exceeds 11 cycles, this sequence must not be used.

Write Operation. C words are written on tape from storage beginning with location Y. When C words have been written, the channel takes its next command from a core location specified by the LCH instruction in the main program or disconnects (and traps if the channel is enabled) if no LCH instruction is waiting to be executed.

In printing or the punching of cards, if the word count is reduced to zero on the 12-row right transmission point and the next command is an IORP or IORT, the end of the present machine cycle (record) will be recognized.

Program Examples

The following examples illustrate the use of data channel commands to read and write tape. The programs, aside from the instructions and command codes, are shown in the octal number system.

Figure 58 shows a program which may be used to read and skip tape records. The first instruction (500) selects the proper channel and tape. The second one loads the first command (1000) into the channel. The command execution proceeds as follows:

LOCATION	INSTRUCTION		COMMENTS
00500	RDS	01201	Select tape 1, channel A
00501	RCHA	01000	Load first command
.	.	.	.
.	.	.	.
.	.	.	.
01000	IOCP	00006 0 03000	Read first six words
01001	IOCPN	00005 2 00000	Skip next five words
01002	IORP	77777 0 03006	Read remaining words in record
01003	IOCD	00000 0 00000	Disconnect

Figure 58. Read and Skip Tape Program

1000 — IOCP 00006 0 03000. This command reads the first six words from tape and places them into core locations starting with 3000.

1001 — IOCPN 00005 2 00000. This command reads, but does not transmit, the next five words, in effect skipping them.

1002 — IORP 77777 0 03006. The next command reads the remaining words in the record into locations starting with 3006. When the record gap is sensed, the next command (IOCD) will disconnect the channel and the tape unit, thus ending the channel program.

Figure 59 shows a program that could be used to write all of core storage on a tape and delay at location 502 until all but the last word has been written.

The first two instructions select the channel and tape unit to be used and load the first channel command. The central processing unit then senses the load channel instruction (LCHF) and the main program will wait until the next-to-last word has been written before loading the second channel command which will write location 77777 and disconnect both the channel and the tape unit.

LOCATION	INSTRUCTION		COMMENTS
00500	WRS	06202	Select tape 2, channel F
00501	RCHF	01000	Load first command
00502	LCHF	01001	Wait until last word, then load channel F
•	•	•	•
•	•	•	•
•	•	•	
01000	IOCT	77777 0 00000	Write locations 00000-77776
01001	IOCD	00001 0 77777	Write location 77777 and then disconnect

Figure 59. Write Tape Program

Channel Trap Operations

ENB — Enable from Y

+0564	F	▨	T	Y
S,1 — 11	12-13	14 — 17	18-20	21 — 35

Description. When this instruction is executed, the contents of location Y determine which signals may cause a trapping operation. Execution of each enable instruction cancels the effect of previous enable instructions. All channels may be disabled (traps will not occur) by executing an enable instruction whose operand contains all zeros.

Indicators. None.

Timing: 2 cycles

Execution. Trapping signals are controlled as follows:

SIGNAL DUE TO	CHANNEL	EFFECTIVE IF A "1" IN POSITION
Channel command or EOF	A	0035
Channel command or EOF	B	0034
Channel command or EOF	C	0033
Channel command or EOF	D	0032
Channel command or EOF	E	0031
Channel command or EOF	F	0030
Channel command or EOF	G	0029
Channel command or EOF	H	0028
Tape check	A	0017
Tape check	B	0016
Tape check	C	0015
Tape check	D	0014
Tape check	E	0013
Tape check	F	0012
Tape check	G	0011
Tape check	H	0010

Execution of a trap will inhibit all further traps until a new enable instruction is executed or a restore-channel-trap instruction is executed. Depression of the reset, clear key, or execution of an RDC instruction, will also disable all channels.

RCT — Restore Channel Traps

+0760	▨	T	▨	14
S, 1 — 11	12 — 17	18-20	21-23	24 — 35

Description. This instruction will allow traps to occur as specified by the previous enable instruction. It cancels the inhibiting effect of an executed trap.

Indicators. Trap Control.

Timing: 2 cycles

Execution. Since the address part of this instruction is a part of the operation code, modification by an index register may change the operation itself.

System Compatibility Operations

ESNT — Enter Storage Nullification and Transfer

- 0021	F	▨	T	Y
S,1 — 11	12-13	14 — 17	18-20	21 — 35

Description. This instruction turns on a half-storage mode indicator, which serves as a protective device for a program being run while using the compatibility feature of the computer.

Indicators. Simulate, trap mode.

Timing: 1 cycle

Execution. With the half-storage mode indicator on, the following events occur: (1) the upper half of storage is made unavailable for reference by a 704 program, (2) index register capacity is halved, and (3) program control is transferred to location Y. The indicator may be reset by depressing the reset, clear, or any of the load keys, execution of any I-O trap (except data channel trap), or execution of an LSNM instruction.

LSNM — Leave Storage Nullification Mode

- 0760	▨	T	▨	10
S, 1 — 11	12 — 17	18-20	21-23	24 — 35

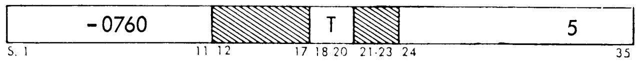
Description. Execution of this instruction returns the computer system to its normal operating capacity by turning the half-storage mode indicator off. If the computer is in its normal operating mode and an LSNM instruction is executed, the instruction will be treated as a no-operation.

Indicators. Simulate.

Timing: 2 cycles

Execution. Since the address part of this instruction is a part of the operation part, modification by an index register may change the operation itself.

ESTM — Enter Select Trap Mode



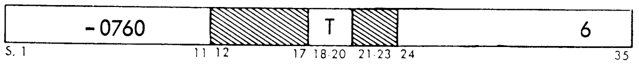
Description. This instruction turns on the select trap mode indicator, and, while in this mode, I-O select and sense instructions are not executed but are trapped. It should be used before entry into a 704 program, so that a 704 instruction will be trapped rather than result in indefinite delays.

Indicators. Simulate.

Timing: 2 cycles

Execution. Instructions that will be trapped include: WEF, BSF, BSR, REW, RDS, WRS, I-O sense, RTT, EOT, redundancy and BOT. The location plus one of the trapped instruction is stored in core storage location 40,000. Program control is transferred to location 40,001. Trapping also turns off the half-storage, select trap, and copy trap indicators. The indicators may also be turned off by: depression of reset, clear, or load keys, or execution of any I-O trap (except a data channel trap). Since the address part of this instruction is a part of the operation code, modification by an index register may change the operation itself.

ECTM — Enter Copy Trap Mode



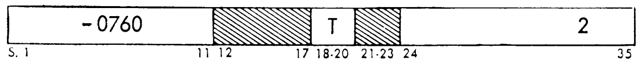
Description. This instruction turns on the copy trap mode indicator. With the indicator on, CPY, CAD, and LDA instructions are trapped instead of being executed.

Indicators. Simulate.

Timing: 2 cycles

Execution. The location plus one of the trapped instruction is stored in core location 40,000 and program control is transferred to location 40,002. The execution of this instruction will also turn off the half-storage, select trap, and the copy trap mode indicators. The copy trap mode indicator may also be turned off by: depression of the clear, reset, or load keys, or the execution of any I-O trap (except a data channel trap). Since the address of this instruction is a part of the operation code, modification by an index register may change the operation itself.

EFTM — Enter Floating Trap Mode



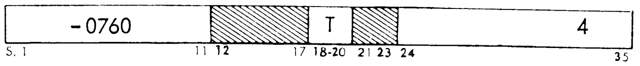
Description. This instruction turns on the indicator for floating-point trap mode. When in this mode, floating-point overflow and/or underflow will cause a trapping operation.

Indicators. MQ overflow.

Timing: 2 cycles

Execution. This mode is the normal operating mode. Floating-point overflow-underflow have the operating characteristics of the standard computer (store location plus one in address 0000 and then transfer to 0010). Since the address part of this instruction is a part of the operation code, modification by an index register may change the operation itself.

LFTM — Leave Floating Trap Mode



Description. This instruction turns off the floating trap mode indicator, giving the computer floating-point overflow characteristics of the standard 704 (turns on the AC or MQ overflow indicators only).

Indicators. MQ overflow.

Timing: 2 cycles

Execution. Depression of the reset, clear, or load keys will return the computer to its normal operating mode (floating trap mode) by turning the floating trap mode indicator on. Since the address of this instruction is a part of the operation code, modification by an index register may change the operation itself.

Commands and Instructions for the IBM 7909 Data Channel

The following instructions and commands are used with the 7909 Data Channel and its attached adapter and input-output devices. Execution timing is not included, because it depends on command organization in core storage and the status of the addressed data channel.

RSCA — Reset and Start Channel A

+0540	F	▨	T	Y
S, 1 – 11	12-13	14 – 17	18-20	21 – 35

Description. On execution of this instruction, the channel is selected and reset and takes its next command from location Y. The instruction is interlocked against channel activity; if the instruction is executed while the channel is busy, its execution is delayed until the channel is in wait status.

Indicators. None.

Execution. If the selected channel is in wait status, the $C(Y)_{S,1-3,19}$ replace the channel operation register, $C(Y)_{3-17}$ replace the word counter, and $C(Y)_{21-35}$ replace the contents of the address counter. In addition, the number Y+1 replaces the contents of the command counter. If the channel is not in wait status, the execution of the CPU program is delayed until the channel executes either a WTR or TWT command.

Instruction codes for other channels are:

INSTRUCTION	CODE	NAME
RSCB	−0540	Reset and Start Channel B
RSCC	+0541	Reset and Start Channel C
RSCD	−0541	Reset and Start Channel D
RSCE	+0542	Reset and Start Channel E
RSCF	−0542	Reset and Start Channel F
RSCG	+0543	Reset and Start Channel G
RSCH	−0543	Reset and Start Channel H

STCA — Start Channel A

+0544	▨
S, 1 – 11	12 – 35

Description. Execution of this instruction is delayed if the channel is not in wait status. If in wait status, the channel is started and takes its next command from the address part of the wait command.

Indicators. None.

Execution. If the channel is in wait status, the command counter is reset and replaced with the contents of the address counter. The channel then executes the command at the location specified by the command counter and increments the command counter by one (adds one to the command counter contents).

Instruction codes for other channels are:

INSTRUCTION	CODE	NAME
STCB	−0544	Start Channel B
STCC	+0545	Start Channel C
STCD	−0545	Start Channel D
STCE	+0546	Start Channel E
STCF	−0546	Start Channel F
STCG	+0547	Start Channel G
STCH	−0547	Start Channel H

SCHA — Store Channel A

+0640	F	▨	T	Y
S, 1 – 11	12-13	14 – 17	18-20	21 – 35

Description. Execution of this instruction causes the specified channel to be selected and that channel's command counter contents to be placed in positions 21-35 of location Y. The channel's address counter contents are placed in positions 3-17 of location Y. Positions S, 1, 2, 18, 19, and 20 of location Y are reserved and their contents cannot be predicted.

Indicators. None.

Execution. The $C(Y)_{21-35}$ are replaced by the contents of the command counter and $C(Y)_{3-17}$ are replaced by the contents of the address counter. The SCHA instruction may be executed at any time, regardless of whether the specified channel is in operation. The command counter may contain the location of the current command or of the next command to be executed.

Instruction codes for other channels are:

INSTRUCTION	CODE	NAME
SCHB	−0640	Store Channel B
SCHC	+0641	Store Channel C
SCHD	−0641	Store Channel D
SCHE	+0642	Store Channel E
SCHF	−0642	Store Channel F
SCHG	+0643	Store Channel G
SCHH	−0643	Store Channel H

ENB — Enable from Y

+0564	F	▨	T	Y
S, 1 – 11	12-13	14 – 17	18-20	21 – 35

Description. When this instruction is executed, the contents of location Y determine which signals may cause a trap operation. Execution of each enable instruction cancels the effect of previous enable instructions. The channel may be disabled (traps will not occur) by executing an enable instruction whose operand contains a zero in the proper position.

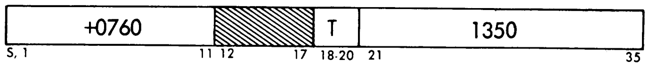
Indicators. None.

Execution. Trapping signals are controlled as follows:

SIGNAL DUE TO	CHANNEL	EFFECTIVE IF A 1 IN POSITION
TWT Command	A	0035
TWT Command	B	0034
TWT Command	C	0033
TWT Command	D	0032
TWT Command	E	0031
TWT Command	F	0030
TWT Command	G	0029
TWT Command	H	0028

Execution of a trap inhibits all further traps until a new enable instruction is executed or a restore channel traps instruction is executed. Depression of the reset or clear key or execution of an RIC instruction also disables the data channel.

RICA — Reset Channel A



Description. This instruction, when executed, causes all conditions in the channel to be reset. The instruction is not interlocked against channel activity. If data transmission is taking place when an RIC occurs, validity of the data already transmitted cannot be guaranteed.

Indicators. None.

Execution. The RIC resets all conditions in the channel to normal initial status and also sends a reset pulse to the adapter. Modification of the address of the RIC may change the operation itself.

Instruction codes for the other channels are:

INSTRUCTION	CODE	NAME
RICB	+0760-2350	Reset Channel B
RICC	+0760-3350	Reset Channel C
RICD	+0760-4350	Reset Channel D
RICE	+0760-5350	Reset Channel E
RICF	+0760-6350	Reset Channel F
RICG	+0760-7350	Reset Channel G
RICH	+0760-10350	Reset Channel H

Other Central Processing Unit Instructions

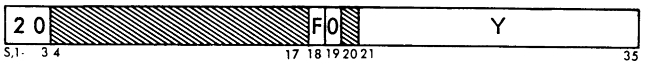
Operation of the following CPU instructions is compatible with operation on the 7607 Data Channel: IOT, RCT, TCNX, and TCOX.

The BTT and ETT instructions always result in a skip, because neither indicator is turned on by the 7909 Data Channel. For the same reason, the TRC and TEF instructions never result in a transfer.

An RDC addressed to a 7909 Data Channel has no effect. Data select instructions (RDS and WRS) or non-data select instructions (BSR, BSF, WEF, REW, RUN, and SDN) addressed to a 7909 cause the 7090 CPU to hang up but have no effect on the 7909 Data Channel.

IBM 7909 Data Channel Commands

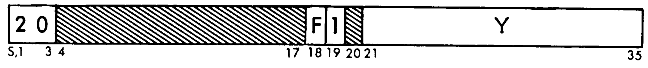
CTL — Control



Description. The control command is decoded in the channel. Information contained in C(Y) is sent to the adapter, starting with the high-order character, and continues until an END signal is received from the adapter. If more than one word location is necessary to transmit all data required by the channel, the next word is taken from location Y+1, etc. This process continues until an END signal is received; the next command is then taken from the storage location following the control command.

Execution. The contents of the address counter are replaced by Y, and the data register contents are replaced by C(Y). When the first data request is received from the adapter, data register contents enter the assembly register and the C(Y+1) replace the contents of the data register. The contents of the assembly register are sent to the adapter, character by character, beginning with the high-order character under control of the adapter. Successive words are sent to the adapter until an END signal is received from the adapter.

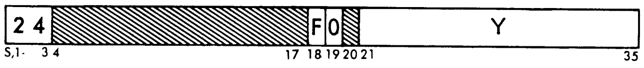
CTLR — Control and Read



Description. This command causes the channel to transmit control information as with the CTL and also prepares the channel for a read operation. When an END signal is received from the adapter, the channel proceeds to the next sequential command (which must be a copy or a TCH to a copy if data transmission is expected). When a copy command is encountered, the channel is placed in read status, and data are transmitted to core storage under control of the copy command.

Execution. Execution is the same as with CTL except that the prepare to read indicator in the 7909 is turned on.

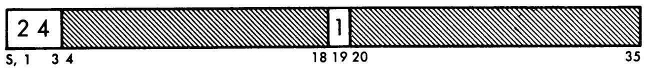
CTLW — Control and Write



Description. This command causes the channel to transmit control information in the same manner as with the control command and also prepares the channel for a write operation. When an END signal is received from the adapter (signaling the end of the order), the channel proceeds to the next sequential command (which must be a copy or a TCH to a copy if data transmission is expected). When the copy command is encountered, the channel is placed in write status, and data are transmitted from core storage to the adapter under control of the copy command.

Execution. Execution is the same as with CTL except that the prepare to write indicator in the 7909 is turned on.

SNS — Sense



Description. Execution of this command places the channel in sense status and then proceeds to the next sequential command (which must be a copy or a TCH to a copy if sense data transmission is expected). When a copy command is encountered, sense information is sent to core storage under control of the copy command.

Execution. Execution of the SNS turns on a sense indicator in the 7909, which causes the adapter to transmit sense data. The channel then proceeds to the next sequential command. A copy (CPYP or CPYD) command is required to provide word count and address information to be used in storing the sense data. If the assembly register is filled before a copy command or a TCH to a copy, a sequence check occurs. If the assembly register is filled before a copy command is encountered, an I-O check results.

A maximum of two sense data words are available from the adapter. Both words can be stored, or it is possible to store only the first word by using a CPYD with a word count of 1. The meaning of each sense data-bit, as placed in core storage from the 7631 File Control is:

FIRST WORD

	BIT POSITION	MEANING IF A 1
	1	Reserved
Summary Bits	3	Program Check
	4	Data Check
	5	Exception Condition
	7	Invalid Sequence
Program Check	9	Invalid Code
	10	Format Check
	11	No Record Found
	13	Invalid Address
	15	Response Check
Data Check	16	Data Compare Check
	17	Parity or Cyclic Code
	19	Access Inoperative
Exception Condition	21	Access Not Ready
	22	Disk Circuit Check
	23	File Circuit Check
	25	Reserved
Status Bit	27	Six-Bit Mode
	28	Reserved
	29	Reserved
	31	Access 0, Module 0
Attention Status	33	Access 0, Module 1
	34	Access 0, Module 2
	35	Access 0, Module 3

SECOND WORD

	BIT POSITION	MEANING IF A 1
	1	Access 0, Module 4
	3	Access 0, Module 5
Attention Status	4	Access 0, Module 6
	5	Access 0, Module 7
	7	Access 0, Module 8
	9	Access 0, Module 9
	10	Reserved
	11	Reserved
	13	Reserved
	15	Reserved
	16	Reserved
	17	Reserved
	19	Reserved
	21	Reserved
	22	Reserved
	23	Reserved

All bit positions (of both sense data words) that are not mentioned in the preceding tables are not used but contain zeros.

The meaning of each sense data-bit as placed in core storage from the 7640 Hypertape control is:

FIRST WORD

BIT POSITION	MEANING IF A 1
1	Operator Required
3	Program Check
4	Data Check
5	Exception Condition
7	Selected
9	Tape
10	Unit
11	Address
13	Selected Drive Not Ready
15	Selected Drive Not Loaded
16	Selected Drive File Protected
17	Operation Not Started
19	Invalid Order Code
21	Selected Drive Busy
22	Selected Drive At BOT
23	Selected Drive At EOT
25	Correction Occurred
27	Channel Parity Check
28	Code Check
29	Envelope Check
31	Overrun Check
33	Excessive Skew Check
34	Track Start Check
35	Not Used

SECOND WORD

BIT POSITION	MEANING IF A 1
1	Selected Drive Read a Tape Mark
3	Selected Drive in EWA
4	Not Used
5	Not Used
7	Read Section Busy
9	Write Section Busy
10	Backward Mode
11	Not Used
13	Drive 0 Attention
15	Drive 1 Attention
16	Drive 2 Attention
17	Drive 3 Attention
19	Drive 4 Attention
21	Drive 5 Attention
22	Drive 6 Attention
23	Drive 7 Attention
25	Drive 8 Attention
27	Drive 9 Attention

Remaining bits do not concern the programmer.

Additional details about sense data from the 7640 include:

Operator Required

This bit is set when operator intervention is required to proceed, as in a failure to execute an instruction or an order because:

1. Selected Hypertape drive is not ready.
2. Selected drive is not loaded.
3. Selected drive is file-protected and an attempt to write is made.
4. The operation was not initiated.

Program Check

This bit is set when the failure to execute an instruction or an order is due to the status of the Hypertape drive and Hypertape control that are under control of the stored program, such as:

1. Invalid operation code.
2. Selected drive is busy.
3. Selected drive is at BOT and a read backward is attempted.
4. Selected drive is at EOT and any order requiring forward tape motion is attempted.

Data Check

This bit is set when an error has occurred in the transmission of data during a read, write, or control operation, as the result of any of the following error conditions:

1. Correction occurred.
2. Channel parity check.
3. Code check.
4. Envelope check. This indicator turns on when: in reading, an uncorrectable error is sensed; in writing, the failure to write a bit or bits is sensed.
5. Overrun. This indicator turns on whenever the computer fails to send or receive a character from the control within the allotted time.
6. Excessive Skew Check. This indicator turns on whenever a bit or bits of a read character fail to fall within the time allocated for that character.
7. Track Start Check. This indicator turns on because of circuit failure in a bit track.

Exceptional Condition

This bit is set when an exceptional condition occurs during the execution of a read or write operation. Exceptional condition does not indicate an error condition. The exceptional conditions are:

1. Tape mark is sensed when reading.
2. Write operation is in progress and end warning area is sensed.

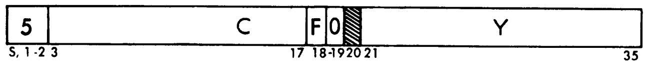
Attention

Ten bits of sense data are used to indicate attention for ten Hypertape drive addresses. These bits are set whenever the corresponding drive signals attention, that is, whenever the drive is placed in ready status or one of the following operations is completed:

Rewind
Unload Cartridge
File Protect
Rewind and Unload
Change Cartridge
Change Cartridge and Rewind

The status data bits also indicate the address of the last selected drive of that channel.

CPYD — Copy and Disconnect



Description. This command, when decoded by a channel not prepared to read or write, causes a sequence check and, thus, a channel interrupt. If the channel is prepared to read or write, this command causes C words to be transmitted between the channel and core storage, starting with location Y. Data transmission continues until C is reduced to zero or an END signal is received by the channel. In either case, the channel read or write select is reset. If, while a CPYD is being executed, an END signal is received before the count is reduced to zero, the channel read or write select is reset, and the channel obtains a new command from the next sequential location.

If the next command is other than a copy, the channel executes that command. If the next command is a copy, the channel interrupts on a program sequence check. The last word transmitted to storage under CPYD control remains in the assembly register if an END signal is received before the word count reaches zero.

If the count for the CPYD goes to zero before the END signal is received, the channel initiates a disconnect but does not get the next sequential command until an END or UNUSUAL END signal is obtained. In general, when operating under CPYD control, the channel does not obtain the next sequential command until either an END (or UNUSUAL END signaling an error) occurs. In the event of an UNUSUAL END signal, an interrupt occurs.

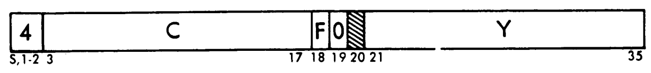
Execution: Read or Sense Operation. Y replaces the contents of the address counter. If the channel is in prepare to read status, this condition is reset and the adapter is signaled to begin transmission of data to core storage. If the read or sense indicator is on from a previous SNS or CPYP, data transmission is continued under control of the CPYD. When the assembly register is full, it is emptied into the data register and access to core storage is requested.

As each word is placed in core storage, the address counter is increased by one and the word counter is decreased by one. If the word count is reduced to zero before an END signal is received, a disconnect is initiated, and the channel obtains its next command when the END signal is received. If a word or partial word has been received, it is stored. If an END signal is received before the word count is reduced to zero, the last word transmitted is stored and the channel gets the next sequential command.

Execution: Write Operation. Y replaces the contents of the address counter. If the channel is in prepare to write status, this condition is reset and the write indicator is turned on. The C(Y) are placed in the data register. When the first data request is received from the adapter, the data register contents are placed in the assembly register and the C(Y+1) replace the contents of the data register.

If the write indicator is on from a previous CPYP command, data transmission is resumed under control of the CPYD. As each word is transmitted to the adapter, the address counter is increased by one and the word counter is reduced by one. Disconnect procedures are the same as for a read or sense operation. A CPYD with a word count of zero causes a disconnect without further data transmission.

CPYP — Copy and Proceed



Description. This command, when decoded by a channel not prepared to read or write, causes a sequence check and channel interrupt. If the channel is prepared to read or write, this command causes C words to be transmitted between the channel and core storage, starting with location Y. END signals from the adapter are serviced, but the channel does not disconnect and data transmission continues until C is reduced to zero. The channel then does not disconnect but obtains the next sequential command. If this command is a TCH, TDC, or a copy, operation is normal and data transmission is resumed. If the next command does not satisfy these conditions, the channel disconnects and interrupts on a sequence error. If an UNUSUAL END occurs, the channel interrupts.

Execution. END signals from the adapter are serviced during a CPYP command. Following the END signal, however, the 7909 signals the adapter to proceed (write, read, or sense). UNUSUAL END conditions cause a channel interrupt. A CPYP command may be followed by a CPYP, CPYD, or TCH or TDC to another copy command.

Use of the CPYP in adapter operation should be carefully controlled. A normal END should never occur during adapter operation using a CPYP command. If word counts are not properly controlled (that is, the total word counts of all CPYP commands on a write operation are greater than the record length), or if the word count is equal to the record length and a CPYP or CPYD with a word count follows, data will be destroyed.

Consider a single record operation on disk where the word count of the record is 100:

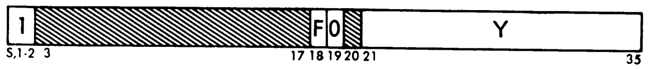
CTLW	DVSR	Verify single record
CPYP	A,,150	Write 150 words
CPYD	B,,10	Write 10 words

When 100 words have been written, the 7631 sends an END signal. The 7909 signals write again, and the remaining 60 words are dumped on top of the first 100. The remaining 40 (of the original 100) are replaced with zeros and no errors are indicated. This condition is possible on all operations except write format. If an attempt is made to exceed the capacity of a format track, a format check results.

It is generally desirable to follow with a write check operation all write operations using CPYP. This assures error detection if a data wrap-around occurs. A routine that may be used to write and write check is:

LOCATION	OPERATION	ADDRESS	COMMENT
	LCC	1	
WR	CTLW	DVSR	Verify single record
	CPYP	A,,150	Write 150 words
	CPYD	B,,10	Write 10 words
	TDC	*+2	Go to write check if control counter not zero
WC	WTR	*	End of write, write check
	CTLW	DWRC	Prepare to write check
	TCH	WR+1	

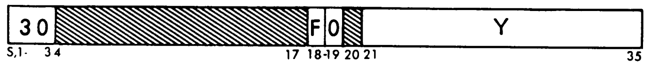
TCH — Transfer in Channel



Description. This command is the transfer command for all channels. When a TCH command is executed, command sequence control is transferred to location Y.

Execution. When a TCH command is executed, the data channel transfers to location Y. The command at location Y is loaded into the data channel and the command counter is increased to Y+1.

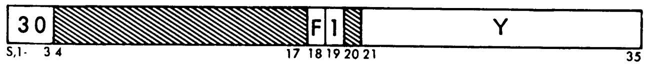
LAR — Load Assembly Register



Description. Execution of this command causes the contents of the assembly register to be replaced by the C(Y). The C(Y) remain unchanged. After execution, the channel proceeds to the next sequential command.

Execution. The contents of Y are sent through the storage bus switches and the data register to replace the contents of the assembly register.

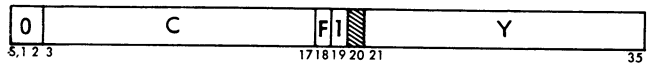
SAR — Store Assembly Register



Description. Execution of the SAR causes the C(Y) to be replaced by the contents of the assembly register. Contents of the assembly register remain unchanged. After execution, the channel proceeds to the next sequential command.

Execution. The assembly register contents are stored in location Y, as specified by the address counter, in the same manner as for a read operation.

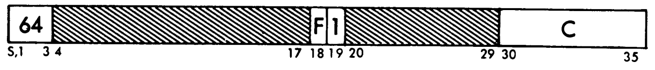
XMT — Transmit



Description. This command causes the C words immediately following the location of the XMT command to be transmitted to C locations starting at location Y. When the C field is reduced to zero and the Cth word has been transmitted, the channel obtains its next command from the location of the XMT command, plus C, plus one. If the initial count field is zero, the XMT command is skipped and the channel proceeds to the next sequential command.

Execution. The command counter is increased by one and the first word is obtained from this location and brought to the data register. The command counter is then increased by one again. The data register contents are stored in location Y. The address counter is increased by one and the word counter is reduced by one. The second word is entered into the data register from the storage location specified by the command counter. This operation proceeds until the word count is reduced to zero. The channel then takes its next command from the location of the XMT command plus C, plus one. The contents of the assembly register remain unchanged. The XMT command may be used to move blocks of data, commands, or entire subroutines from one area of core storage to another area.

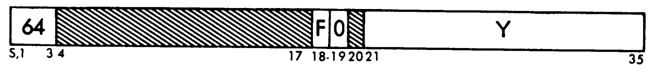
LCC — Load Control Counter



Description. This command causes the contents of the channel control counter to be replaced by the six low-order positions of the count field of the LCC command. The channel then proceeds to the next sequential command. If the LCC is indirectly addressed, the contents of the control counter are replaced by the six low-order bits contained in the location specified by positions 21-35 of the LCC command.

Execution. The control counter is reset. The contents of positions 12-17 of the address counter are placed in positions 1-6 of the control counter. The channel then proceeds to the next command as specified by the command counter.

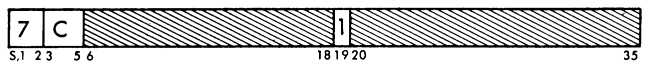
TDC — Transfer and Decrement Counter



Description. On execution of this command, the contents of the six-bit channel control counter are examined. If the contents are not zero, the counter is reduced by one and control is transferred to location Y. If the counter contents are zero, the channel proceeds to the next sequential command, leaving the counter contents unchanged.

Execution. If the control counter contains a count of zero, the channel proceeds to the next command as specified by the command counter. If the control counter is not zero, it is decreased by one. The command counter is reset and replaced by the contents of the address counter. The channel then takes the next command from location Y.

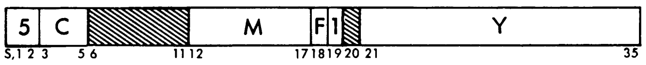
ICC — Insert Control Counter



Description. When the count field (C) of the ICC is not zero, this command causes the C field to specify one of the six characters in the assembly register to be replaced by the contents of the control counter. The remaining five characters are not affected. If C is zero, the sixth character of the assembly register is replaced by the contents of the SMS status indicators. In either case, the channel proceeds to the next sequential command after execution of the ICC. An ICC with a C field of seven functions as a no operation. The contents of the assembly register remain unchanged.

Execution. Word counter positions 3, 4, and 5 are decoded to specify a character of the assembly register. The selected character is reset and replaced by the control counter if the C field is one through six or by the SMS status indicators if the C field is zero.

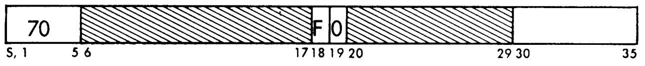
TCM — Transfer on Condition Met



Description. When C is not zero, this command causes C to specify one of the six characters in the assembly register for comparison against the contents of the mask (M) field. If a bit-for-bit comparison is achieved, the channel executes a transfer to location Y. If the comparison is not achieved, the channel proceeds to the next sequential command. If C is zero, the channel check condition register is compared against M. Transfer conditions for the comparison are as previously stated. When indirect addressing is used, control is transferred to the indirectly addressed location when the condition is met. If C is equal to seven, the result depends on mask contents. If all bits in positions 12-17 of the TCM are zero, the channel executes a transfer to location Y. Otherwise, the channel proceeds to the next sequential command.

Execution. Word counter contents (position 3, 4, and 5) are decoded to find the count. Assembly register contents are divided into six characters with the first character in positions S, 1-5 and the last character in positions 30-35. If the count field is a value one through six, one of the characters from the assembly register (specified by C) is compared with the M field of the TCM. If C is zero, the six-position check condition register is compared with M. If C is seven, a six-bit character of all zeros is compared with M. If a bit-for-bit comparison is achieved, the channel transfers control to location Y. The command counter is reset and replaced with the contents of the address counter. If a comparison is not achieved, the channel proceeds to the next sequential command. The contents of the assembly register and the check condition register remain unchanged.

SMS — Set Mode and Select



Description. Execution of this command causes the contents of positions 30-35 of this command to set or reset specific status indicators as follows:

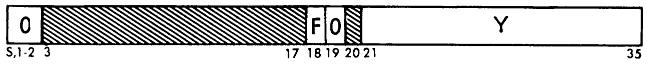
BIT	FUNCTION
30*	Read Backward
31*	BCD Mode
32	Inhibit Unusual End Signals
33	Inhibit Attention 1 Signals
34*	Inhibit Attention 2 Signals
35*	Select 2 (1 is selected when reset)

*Optional Features

Bits 34 and 35 apply to the data channel switch feature described in "Optional Features."

In all cases, the presence of the bit causes the status indicator to be set and the function to be enabled; absence of the bit resets the status indicator and disables the function. Machine and power-on resets also reset the indicators. With indirect addressing, the SMS command status indicators are set or reset with bits 30-35 of the location specified by bits 21-35 of the indirectly addressed command. After execution of the SMS, the channel proceeds to the next sequential command.

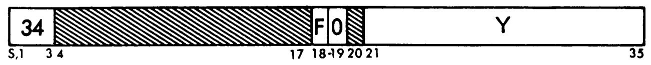
WTR — Wait and Transfer



Description. When this command is decoded, the channel stops operation and may be thought of as waiting. The channel location counter contains the location of the WTR command. When the channel is told to start, it takes its next command from the location specified by Y of the WTR command. If an interrupt occurs while the channel is in wait status, return from the interrupt program by means of a LIP command puts the channel in wait status.

Execution. Execution of this command forces the channel to wait. The channel may be restarted by either an RSC or STC command or by an interrupt. The command counter is not changed, and the address counter contains Y. If the WTR is indirectly addressed, the address counter contains the contents of the address portion of location Y.

TWT — Trap and Wait



Description. Upon decoding a TWT command, the channel suspends operation until either a reset and start or start channel instruction is executed by the CPU, depending on conditions described below. If the channel is enabled for control word traps, the channel causes the CPU to trap to a fixed location. Particulars concerning this trap are described in "Data Channel Trap."

If the channel is enabled and encounters a TWT command, start channel instructions are ignored until the trap is executed or a reset and start channel instruction is executed. If the channel is not enabled, either a reset and start or start channel instruction resets the trap and causes the channel to resume operation.

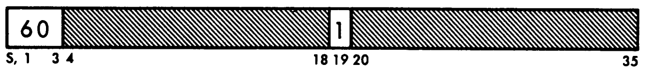
Channel interrupt signals are remembered but not executed until the channel brings in a command other than TWT. (An RSC resets these stored interrupt signals.) After the channel has stopped operation as a result of a TWT, the channel command counter contains the location of that command.

Assume that B is the location where the instruction counter contents are stored when a trap occurs on this particular channel and that CPU control is transferred to B+1. SUB is the entry point for the subroutine that the channel requests the CPU to execute.

COMMAND	ADDRESS	COMMENT
XMT	B+1,,1	Moves the following TRA to location B+1.
TRA	SUB	
TWT	Y	Transfers control to CPU at location B+1.

Execution. When the TWT is decoded, the wait indicator is turned on, the channel trap demand is initiated, and the channel waits. The command counter is not changed and the address counter contains Y. If the TWT is indirectly addressed, the address counter contains the contents of the address portion of location Y.

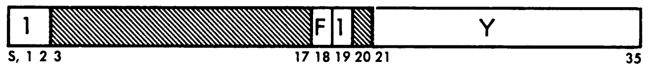
LIP — Leave Interrupt Program



Description. This command causes the channel to transfer control to the location contained in the address part of the channel's fixed "interrupt-to" location. The channel command counter is set to the value in the address portion of this fixed location. Execution of the LIP also cancels the inhibiting effect of a previous interrupt.

Execution. The interrupt and check condition indicators are reset and the contents of the address portion of the fixed interrupt-to location are entered in the command counter. The contents of the location specified by the command counter are loaded into the operation register, word counter, and address counter. The command counter is stepped to the location of the next command.

LIPT — Leave Interrupt Program and Transfer



Description. Execution of the LIPT command cancels the inhibiting effect of a previous interrupt and transfers channel control to location Y. Use of the LIPT permits returning from the interrupt subroutine to a program location other than the interrupt address.

Execution. When an LIPT command is executed, the interrupt and check condition indicators are reset and the channel proceeds to location Y for its next command. The command located at Y is loaded into the channel and the command counter is increased to Y+1.

Programming Example of LIPT Command

LOCATION	INSTRUCTION OR COMMAND	ADDRESS	COMMENT
100	XMT	RSTRT,,1	Store return address in
101	PZE	102	indirect address restart location.
102	CTLR	DVTN	Read 30 words using
103	CPYP	BEGIN,,10	track with no ad-
104	CPYD	END,,20	dresses.

RSTRT is a location used to store restart addresses. The XMT command stores PZE 102 at location RSTRT. Assume that an I-O check occurs during the CPYP command. The channel initiates a disconnect and interrupts. The command counter contains 104, the location of the next command. If the error routine decides to try again, an LIP command cannot be conveniently used, because the channel leaves the interrupt routine and transfers to location 104. A sequence check would occur followed by another interrupt.

If the interrupt routine is ended with an indirectly addressed LIPT command (LIPT RSTRT,4), rather than a LIP, the channel returns to the CTLR command and retries the failing section of the program.

IBM 7909 Data Channel Command Bit Configurations

S	1	2	3	19	Command	
0	1	0	0	0	CTL	Control
0	1	0	0	1	CTLR	Control and Read
0	1	0	1	0	CTLW	Control and Write
0	1	0	1	1	SNS	Sense
0	1	1	0	0	LAR	Load Assembly Register
0	1	1	0	1	SAR	Store Assembly Register
0	1	1	1	0	TWT	Trap and Wait
1	1	0	0	1	LIP	Leave Interrupt Program
1	1	0	1	0	TDC	Transfer and Decrement Counter
1	1	0	1	1	LCC	Load Control Counter
1	1	1	0	0	SMS	Set Mode and Select
0	0	0	▨	0	WTR	Wait and Transfer
0	0	0	▨	1	XMT	Transmit
0	0	1	▨	0	TCH	Transfer in Channel
0	0	1	▨	1	LIPT	Leave Interrupt Program and Transfer
1	0	0	▨	0	CPYP	Copy and Proceed
1	0	1	▨	0	CPYD	Copy and Disconnect
1	0	1	▨	1	TCM	Transfer on Condition Met
1	1	1	▨	1	ICC	Insert Control Counter

IBM 7631 File Control Order Bit Configurations

2	3	4	5	8	9	10	11	Num Code		Order
1	0	1	0	1	0	1	0	00	DNOP	No Operation
1	0	1	0	0	1	0	0	04	DREL	Release
1	0	1	0	1	0	0	0	08	DEBM	Eight-Bit Mode
1	0	1	0	1	0	0	1	09	DSBM	Six-Bit Mode
1	0	0	0	1	0	1	0	80	DSEK	Seek
1	0	0	0	0	0	1	0	82	DVSR	Prepare to Verify (single record)
1	0	0	0	0	0	1	1	83	DWRF	Prepare to Write Format
1	0	0	0	0	1	0	0	84	DVTN	Prepare to Verify (track with no addresses)
1	0	0	0	0	1	0	1	85	DVCY	Prepare to Verify (cylinder operation)*
1	0	0	0	0	1	1	0	86	DWRC	Prepare to Write Check
1	0	0	0	0	1	1	1	87	DSAI	Set Access Inoperative
1	0	0	0	1	0	0	0	88	DCTA	Prepare to Verify (track with addresses)
1	0	0	0	1	0	0	1	89	DVHA	Prepare to Verify (home address)

* Optional Feature

IBM 7640 Hypertape Control Order Bit Configurations

2	3	4	5	8	9	10	11	Numeric Code	Mnemonic Code	Orders
1	0	1	0	1	0	1	0	00	HNOP	No Operation
1	0	1	0	0	0	0	1	01	HEOS	End of Sequence
1	0	1	0	0	0	1	0	02	HRLF	Reserved Light Off
1	0	1	0	0	0	1	1	03	HRLN	Reserved Light On
1	0	1	0	0	1	0	1	05	HCLN	Check Light On
1	0	1	0	0	1	1	0	06	HSEL	Select
1	0	1	0	0	1	1	1	07	HSBR	Select for Backward Reading
0	0	1	0	1	0	0	0	28	HCCR	Change Cartridge and Rewind
0	0	1	1	1	0	1	0	30	HRWD	Rewind
0	0	1	1	0	0	0	1	31	HRUN	Rewind and Unload Cartridge
0	0	1	1	0	0	1	0	32	HERG	Erase Long Gap
0	0	1	1	0	0	1	1	33	HWTM	Write Tape Mark
0	0	1	1	0	1	0	0	34	HBSR	Backspace
0	0	1	1	0	1	0	1	35	HBSF	Backspace File
0	0	1	1	0	1	1	0	36	HSKR	Space
0	0	1	1	0	1	1	1	37	HSKF	Space File
0	0	1	1	1	0	0	0	38	HCHC	Change Cartridge
0	0	1	1	1	0	0	1	39	HUNL	Unload Cartridge
0	1	0	0	0	0	1	0	42	HFPN	File Protect On

Systems Program Compatibility

704 Programs on 7090 System

The compatibility II program makes possible the execution of programs written for a 704 system on the 7090 system. The compatibility program simulates 704 input-output operations through use of the storage-nullification, input-output select trap, and copy trap modes. The program requires no modification of the 7090 on which it is used; however, it cannot be used on a 7090 system with less than 8192 words of core storage. Also, if the last location of a 704 high-end loader contains a copy, the instruction counter (40000) is stored at address 40000 and the subsequent ESNT results in a halt.

Compatibility II executes a leave floating-point trap mode (LFTM) instruction before entering a 704 program, in order that overflow will function as on a standard 704. Computations are not affected in any other way. The program is designed to use all of the upper half of storage. Some of these locations are used for storing the program itself; the rest are used as an input-output buffer between the 704 program and tape or card units, and, if required, for simulating magnetic drums. All locations of the upper half of storage not used for the compatibility program instructions or for drum simulation are used for the input-output buffer.

Before a 704 program can be processed, a control card must be read. This control card indicates the 7090 equivalents of the 704 tapes used in the processing of the 704 program. After the control card is read, the compatibility program enters select trap, copy trap, and storage nullification modes and simulates a load card, load tape, or load drum operation depending on the setting of the console entry keys. Until the completion of the 704 program, input-output operations are simulated through use of the select trap and copy trap modes of operation. (All other instructions are compatible with the 7090.)

See IBM 709 Data Processing System Bulletin, J28-6039 for a complete explanation.

709 Programs on 7090 System

Programs written for the 709 may be run on the 7090 without modification or sacrifice in efficiency and still take advantage of the increased system speed. There are, however, differences which are potential areas of incompatibility. These are:

1. Read or write drum instructions will be trapped if an ESTM instruction has been executed. CPY, CAD,

and LDA instructions will be trapped if an ECTM is executed. When the instruction is not trapped, the I-O indicator will be turned on and the instruction will be treated as a no-operation. CRT instructions may be trapped but will always give an I-O check indication.

2. It is usually possible to simulate the drum on the 7090 system by using the 704 compatibility feature. If, however, the 709 program uses data channel traps, difficulty may be encountered if traps occur while the compatibility program is being executed. This will result in returning control to the 709 program without allowing the 7090 to set proper operating modes.

3. The change in the ratio of compute speed to input-output speed will affect programs that depend upon computed delays for satisfactory operation. Since the 7090 is faster than the 709, difficulty will occur only in cases where "shrinkage" of a delay loop can cause trouble. For instance, a program may assume that certain storage locations may be changed *n* machine cycles after a write tape instruction. Thus, if that area is used without making a test, the 709 and 7090 may not write the same data.

4. To achieve compatibility, the 7090 system must have the same complement of data channels, tape units, and card equipment. Further, these units must be arranged in the same way with regard to addressing.

5. Since magnetic tape may be recorded at two densities on the 7090, provision must be made to set up the tape units for the particular density required. This may be done by use of the change density switch located on each tape unit or by the SDN instruction.

7090 Programs on 709 System

To run a 7090 program on a 709, the same general precautions observed with a 709-to-7090 program must be used. Instructions pertinent to the 7090 only (instructions referring to channels G and H) will cause the 709 to hang up.

Inasmuch as the computed delays will be much longer when 7090 programs are run on a 709, trouble will be encountered whenever some critical timing may not be exceeded. For example, if the maximum allowable number of 7090 instructions are executed between select and reset and load instructions, an I-O check will result when this program is run on the 709. This is essentially the inverse of the problem encountered when 709 programs are run on a 7090. Again, the size and configuration of the systems must be the same. It is possible, however, to write programs for 7090 systems which are not attainable with the 709. This will occur because a 7090 data channel has more than eight tape units, the 7090 system uses channels G and H, and card equipment on the 7090 may appear on channels B, D, or F.

Since the 709 does not have dual-density tape units, all programs will produce low density tape as output and must have low density tape as input.

Interrupt

The 7909 Data Channel is capable of interrupting its stored program independently of the main computer and other data channels. This operation is separate and distinct from a data channel trap—which interrupts the CPU—and represents another significant departure from 7607 Data Channel operation. On recognition of an interrupt condition, the 7909 channel stores the contents of the command and address counters in a fixed location and then executes the command located in another fixed location. This process is termed *interrupt.*

If the 7909 channel is to be diverted from normal command execution sequence, the command in the fixed location must be one that will change the contents of the command counter (TCH, LIPT, or successful TDC or TCM). If this command is other than a successful transfer, the channel executes it and resumes operation at the location immediately following the location where the interrupt occurred. If the command at the fixed location is a WTR or TWT, the channel suspends operation as described in the channel command section, but the command counter contains the location plus one of the command responsible for the interrupt. The channel interrupt locations are assigned as follows:

CHANNEL	STORE COMMAND COUNTER AT	OBTAIN NEXT COMMAND FROM
A	00042	00043
B	00044	00045
C	00046	00047
D	00050	00051
E	00052	00053
F	00054	00055
G	00056	00057
H	00060	00061

When the 7909 interrupts, the command and address counters are automatically stored in the assigned fixed core storage location. The address counter is stored in positions 3-17 and the command counter in positions 21-35 of this location.

Interrupt conditions are stored in a six-position register in the data channel and may be examined with the TCM command. Any combination of interrupt conditions causes an interrupt; however, once interrupted the channel is placed in interrupt mode and further attempts to set the interrupt condition or to interrupt are inhibited. The channel remains in interrupt mode until an LIP or LIPT command is executed by the channel or an RIC instruction is executed by the CPU. If a channel is in interrupt mode and an RSC instruction is executed by the CPU before the channel executes a LIP or LIPT command, the interrupt condition register is reset but the channel remains in interrupt mode. An LIP or LIPT command or a RIC instruction is the only program means available to cause the channel to exit from interrupt mode and become receptive to further interrupt conditions.

Interrupts are also inhibited if channel trap is in process on that channel. This inhibiting persists until either an RSC or STC instruction (depending on whether the channel was enabled) is executed by the CPU (see "TWT Description").

Interrupt Conditions

Interrupt indications are stored in a six-position register in the data channel. The contents of this register may be examined by the TCM command. The positions of the register and the conditions they reflect are:

POSITION	CONDITION
1	Input-Output Check
2	Sequence Check
3	Unusual End
4	Attention 1
5	Attention 2
6	Adapter Check

Input-Output (I-O) Check

This condition occurs when the channel fails to obtain a storage reference cycle in time to satisfy demands of the attached I-O device. The condition is also monitored in the CPU and reflected by the I-O check light. The condition of the light may be tested by the CPU, using the IOT instruction. The input-output test (IOT) instruction execution turns the I-O check light off in the CPU but will not affect the 7909 I-O check indicator. The channel I-O check idicator is turned off when an LIP or LIPT command is executed or when the CPU executes an RSC or RIC instruction.

The channel I-O check indicator being on indicates one of the following conditions:

1. During a write or control operation, the channel data register has not been loaded with a word from

core storage by the time its contents are to be sent to the adapter.

2. During a read or sense operation, the channel data register has not been stored by the time new data are completely assembled in the assembly register.

When an I-O channel check occurs, the adapter is disconnected and an interrupt occurs when the END signal is received from the adapter. The command counter contains the location plus one of the present command. The address counter contains the location plus one or two of the last word transmitted if the operation was a write or control, or the location plus one of the last word transmitted if the operation was a read or sense.

If an I-O check occurs while the channel is in interrupt mode, the I-O check is not recognized and is not saved.

Sequence Check

A sequence check indicates an invalid sequence of channel commands. Improper command sequences occurring while the channel is in interrupt mode may cause the 7909 to hang up. If a sequence check occurs during data transmission, the adapter is logically disconnected and the interrupt occurs when the END signal is received. In general, data transmission (read, write, or sense operations) may be started by one of the following sequences: a CTLR followed by a CPYP, a CTLW followed by a CPYP, or an SNS followed by a CPYP. Once transmission has been started with a CPYP, it must be ended with a CPYD. Between the first CPYP and the CPYD, transfers are possible but only three commands (CPYP, TCH, or TDC) are permitted.

The following conditions cause a sequence check and a channel interrupt:

1. If a CTLW, CTLR, or SNS is followed by CTL, CTLW, CTLR, WTR, TWT, or SNS.

2. If an SNS or CPYP is followed by any command other than a CPYP, CPYD, TCH, or TDC.

3. If a TCH or TDC following an SNS or CPYP transfers control to any command other than a CPYP, CPYD, TCH, or TDC.

4. If a CPYP or CPYD has not been properly preceded by a CTLW, CTLR, or SNS.

Unusual End

An UNUSUAL END indicates an error condition recognized by the adapter. This condition causes an immediate interrupt. The reason for the UNUSUAL END may be determined by sensing the adapter error indicators (see "SNS — Sense Command").

UNUSUAL END interrupts may be disabled by the SMS command with a 1-bit in position 32. If UNUSUAL END signals are inhibited and an UNUSUAL END is received, it is treated as a normal END but the UNUSUAL END indicator is set. A later SMS with a 0-bit in position 32 does not reset the indicator and, if not reset by other means — such as an LIP or LIPT command or an RIC instruction — the next END signal (normal or unusual) received from the adapter causes an interrupt.

If an I-O, sequence, or adapter check occurs during a data transmission operation, the operation is immediately ended with a STOP signal and an interrupt occurs when the END signal is received. If an UNUSUAL END occurs when transmission is ended, this condition is recognized. The channel does not interrupt twice but has both error indications available for examination during the interrupt routine. Data read or written during an operation that ended with an interrupt may be incomplete or invalid.

Attention Conditions

This is a signal indicating a change in status of the attached input-output device. During disk operations, an attention signal is generated when an access mechanism has completed a seek operation. The particular access mechanism that generated this indication may be determined from sense data.

The single attention indicator in the adapter, common to all access mechanisms, is reset when the 7909 interrupts. The individual access bits are reset by giving a read, write, or control command to the individual access address.

There are two attention indicators in the 7909. Attention 1 indicates a signal from the device attached to position 1 of the data channel switch feature; attention 2 indicates a signal from the device attached to position 2.

Either or both attention interrupts may be disabled with the SMS command. If attention interrupts are inhibited, the status indicator is set but no interrupts occur and no attention response is sent to the adapter. When an SMS that enables an existing attention indicator is executed, the interrupt occurs at termination of the SMS, and the attention response is sent to the adapter.

Attention interrupts are serviced only at the logical termination of the command during which they occur. The logical termination of a read or write operation is the disconnect resulting from a CPYD. Attention signals occurring while the channel is in interrupt mode do not set the status indicators; however, a second interrupt, to service the adapter attention signal, occurs as soon as the channel leaves the interrupt mode. In this case, the channel executes the command following the LIP or LIPT command before interrupting on the second attention signal.

Adapter Check

An adapter check indicates an error recognized by the 7909 and does not necessarily indicate an adapter malfunction. Conditions causing an adapter check are:

1. Circuit failure occurs in the 7909 assembly ring or character ring (Figure 62).
2. The character rate of the attached I-O device exceeds the capability of the channel.
3. The adapter is not operational. This type of indication occurs if power is off on the adapter and an attempt is made to read, write, control, or sense. On shared disk storage systems, this indication occurs if an attempt is made to read, write, control, or sense and the adapter is in operation on the sharing system.

If an adapter check occurs while the adapter is selected, the adapter is logically disconnected and the interrupt occurs when the END signal is received.

Input-Output Components

Magnetic Tape Units

As many as ten IBM 729 II, IV, V or VI Magnetic Tape Units can be connected to each 7607 Data Channel on the 7090 system. Tape units may be intermixed, as to type, on the 7090 without addressing changes.

Character Alteration in BCD Mode

As six-bit BCD characters are read from magnetic tape, the zone bits of some of the characters are altered. This alteration is performed so that the digits 0-9 and the characters A-Z are represented in core storage by six-bit binary numbers of increasing magnitude. The alteration of these zone bits is:

	IN CORE STORAGE	ON TAPE
CHARACTERS	B A	B A
Numeric	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; that is, by their exact values as binary integers. Thus, the zone part of the digits is 00. The number zero is represented on magnetic tape by the bit configuration 001010. This representation is automatically altered to 000000 during reading in the BCD mode.

During writing in the BCD mode, the alteration procedure is reversed so that the storage BCD characters are transformed to the BCD tape format by the tape control. In writing, the tape control automatically performs the modifications described but does not check whether the six bits being transmitted form a legitimate BCD character. Thus, if binary numbers are written in the BCD mode, both a pure zero and the number 10 (001010) are recorded on the tape as the BCD zero character (001010). Also, the integer 15 (001111) is identical to the BCD tape mark, signifying an end-of-file condition. Therefore, random binary data should not be recorded in the BCD mode.

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

Detailed magnetic tape unit descriptions are in the *Reference Manual, IBM Magnetic Tape Units,* Form A22-6589.

Disk Storage

The IBM 1301 Disk Storage and IBM 7631 File Control (Figure 60) are available as an optional feature on all IBM 7090 Data Processing Systems. As many as five

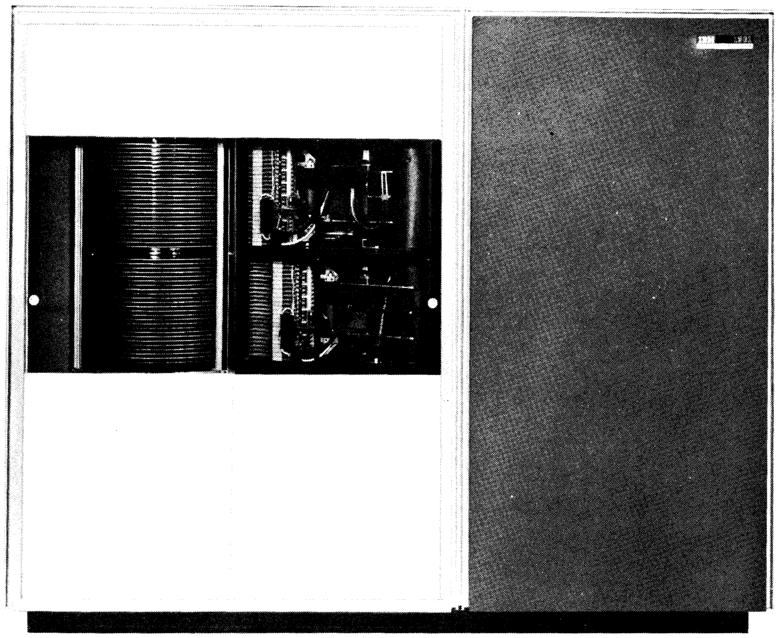


Figure 60. IBM 1301 Disk Storage

disk storage units may be attached to one or two file control units. The 7631 is available in four models:

Model 1: Used with 1410 Data Processing Systems.

Model 2: Used with 7070, 7074, 7080, 7090, and 7094 Data Processing Systems.

Model 3: Used when disk storage is to be shared by a 7000 series system and a 1410 system.

Model 4: Used when disk storage is to be shared by any two 7000 series systems.

Disk storage units may be attached to a 7090 system as shown in Figure 61. Other arrangements may be made by using two file control units and two 7909 data channels. For example, three disk storage units may be attached to one file control (and its 7909 data channel) while one or two disk storage units are attached to another file control and its 7909 data channel. Normal data channel addresses are used and, therefore, no more than eight data channels (7607 and 7909) may be used with any one 7090 system.

Each of the five possible 1301 units may have two modules. Within a 1301, the lower module is numbered 0 and the upper module is numbered 1. To the computer, the ten modules are numbered 0 through 9. The access mechanism of each module is numbered 0. A combination of these two numbers is used to select a module and an access. For example, the upper module of the third 1301 in a system is addressed by the number 05.

In writing a format track, inter-record gaps are 12 characters long. On the format track, address and data areas must have the exact character count that will be recorded in the data area, because zero characters are written on the data track for every unused character position on the format track.

Data leave and enter the IBM 7090 as six 6-bit characters in both six-bit or eight-bit mode. When the system is writing in eight-bit mode, two zero bits are added to each character as it is recorded on the disk surface. When the system is reading in eight-bit mode, the same two bit positions are dropped in the data channel and only the six-bit character is read into storage.

Transmission of data to and from disk storage is accomplished by data channel commands. Data transmission is similar to that used by the 7607 channels, except that read and write operations are decoded in the 7909 channel rather than in the central processing unit. Disk control operations that do not require data transmission are accomplished by sending an order to the file control, where it is decoded and executed. A complete description of these orders and their operation is in the *General Information Manual, IBM 1301 Disk Storage with IBM 7000 Series Data Processing Systems,* Form D22-6576-2.

Data Flow

Figure 62 is a simplified flow chart showing 7909 Data Channel registers and data switches concerned with data flow.

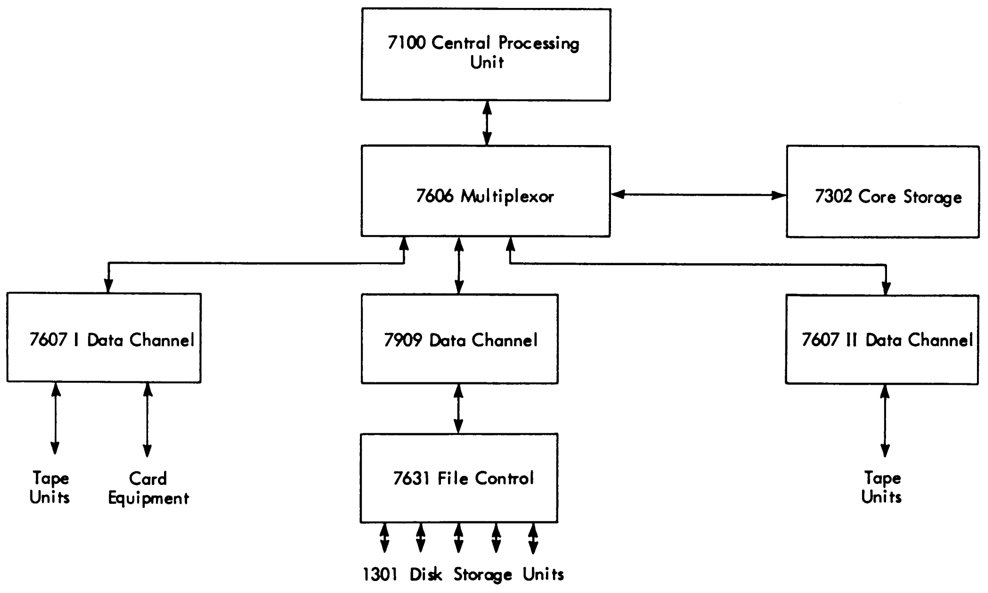


Figure 61. IBM 7090 System with IBM 1301 Disk Storage

Storage Bus Switches: These 36-position switches provide the data path to and from the 7606 Multiplexor for data and command entry into the 7909.

Data Register: This 36-position register is a buffer register for data flow between core storage and the assembly register. During a write or control operation, the data register is loaded with the next data word to be sent to the adapter. On a read or sense operation, the input data are kept in the data register until the data can be placed in core storage.

Assembly Register: This 36-position register assembles and disassembles data passing between the 7909 and the adapter.

Channel Address Switches: This 15-position switch provides the 7606 with address information. The next word needed by the 7909 is obtained by directing the address counter to the channel address switch if data are to be transmitted or by directing the command counter to the channel address switch when a new command is required.

Operation and Control Registers: During a command word cycle, the storage bus switches are directed to the operation register, word counter, and address counter. Positions S, 1, 2, 3, and 19 enter the operation register. These five bits are decoded and provide the 7909 with its next command. Positions 21-35 enter the address counter. During data operations, the address counter contains the location of the next data word. During transfer type commands (TCM, TDC), the address counter contains the location of the next channel command. Positions 3-17 enter the word counter, which is used to control the number of words passed between the 7909 and core storage.

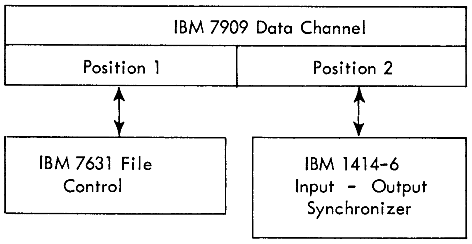
Command Counter: The 15-position command counter contains the location of the next 7909 command. The first operation performed during all command execution—except WTR and TWT—is to step the command counter to the next sequential command location. The command counter is reset and reloaded by execution of an RSC, TCH, LIP, LIPT, or successful TCM or TDC command.

Character Ring: The character ring completes a cycle for each character transmitted. Its main use is to synchronize character-bit transmission.

Assembly Ring: The assembly ring is a character counter and gates data into or out of the assembly register as required. During data operations, data are sent to or received from the adapter, one 6-bit character at a time, through the character switches.

IBM 7909 Data Channel Switch Optional Feature

The IBM 7909 Data Channel Switch permits simultaneous attachment of one or two input-output control units to one 7909 Data Channel. One IBM 7631 File Control and one IBM 1414-6 Input-Output Synchronizer may be attached to one data channel as shown in the flow chart:



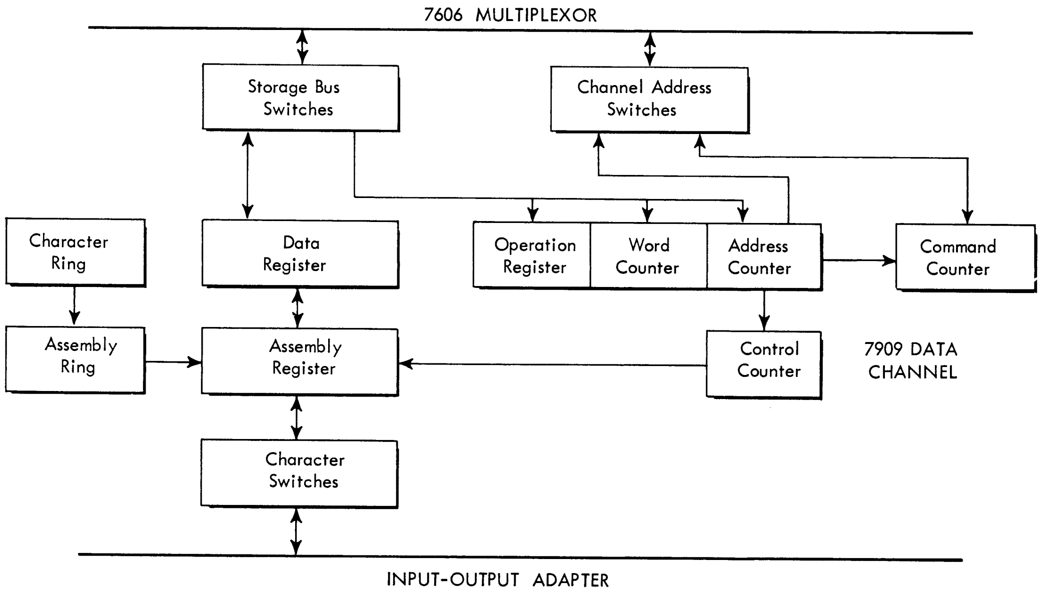


Figure 62. IBM 7909 Data Channel Data Flow

When the 7909 is equipped with the channel switch, data transmission occurs between the 7909 and one of the control units. Attention signals, however, are monitored from each control unit simultaneously. The 7909 is then able to select the control unit and determine priority of attention requests by means of its own stored program.

UNUSUAL END and I-O check signals apply only to that control unit currently selected. Indicators are included to record attention signals from the non-selected control unit and to denote which control unit is currently selected. Attention interrupts occur on signal from either or both of the control units and are subject to the same limitations (described earlier) as without this feature.

IBM 7909 Data Channel BCD Translation Optional Feature

This feature provides automatic BCD translation for information transmitted between the input-output adapter and the 7909 Data Channel. After execution of an SMS command (with a 1-bit in position 31) data transfers between the 7909 and the adapter are translated as shown in the table that follows. Control and sense data are not translated. An SMS command with a 0-bit in position 31 returns data transfer mode to binary.

This feature allows the IBM 7090 system to share data recorded in disk storage with other systems when using a shared disk file. Translation of data is shown in the following table with the characters divided into their bit configurations; each character is shown as it appears in core storage and on disk storage:

Core Storage						Disk Storage					
B	A	8	4	2	1	B	A	8	4	2	1
0	0	0	0	0	0	0	0	1	0	1	0
0	0	1	0	1	0	0	1	0	0	0	0
0	0	(other)				0	0	(no change)			
0	1	(any				1	1	(no change)			
1	0	(any)				1	0	(no change)			
1	1	0	0	0	0	0	0	0	0	0	0
1	1	(other)				0	1	(no change)			

With the translation feature enabled, binary data may be written in BCD format and recovered by reading in the BCD mode.

Home addresses and record addresses may be written using BCD format. This automatically provides the BCD format required in all address areas. Note, however, that subsequent attempts to verify an address written in BCD format will fail unless the adapter orders are program-modified to conform to disk storage BCD codes. Information sent to the adapter during a control operation is not translated whether in BCD mode or not. This is a restriction only where portions of the address are alphabetic characters. Numeric addresses must always be expressed in BCD format.

IBM 7340 Hypertape Drive

The IBM 7340 Hypertape Drive (Figure 63), with the IBM 7640 Hypertape Control, introduces a new concept in magnetic tape devices. Advantages of the 7340 tape system are:

Character Rate: As many as 170,000 alphameric characters or 28,330 words per second.

Reel Capacity: In some applications, more than twice that of 729 IV reels (recorded at high density) even though a reel of 7340 tape is 600 feet shorter.

Cartridge: Machine and file tape reels contained in a cartridge; result is faster loading and unloading of tape without manual threading of the tape.

Read Backward: Allows a recorded tape to be read backward and eliminates necessity of a rewind operation between successive reads of the same recorded tape.

Faster Access: Average access to records in 4.2 milliseconds.

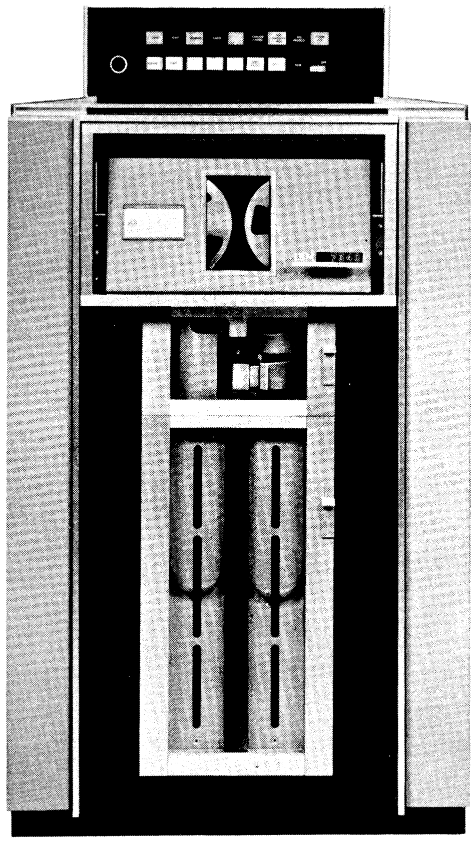


Figure 63. IBM 7340 Hypertape Drive

Checking: Automatic detection of all data errors. Automatic correction of all single-bit errors and most double-bit errors.

File Protection: Cartridge file-protect device under program control.

A tape character consists of the information recorded in a bit-wide column across the ten tracks, perpendicular to the edges of the one-inch tape used by the IBM 7340 Hypertape Drive. When the 7340 is used with the 7090 system, eight of these tracks are used, six for data recording and two for error detection and correction. The remaining two tracks are not used with the 7090 system. Tape track assignments on the 7340 for both BCD and binary codes are:

	BCD Code	Binary Code						
Check Bits	C0	C0	C0	C0	C0	C0	C0	Check Bits
	C1	C1	C1	C1	C1	C1	C1	
Not used with 7090	--	--	--	--	--	--	--	Not used with 7090
	--	--	--	--	--	--	--	
Data Bits	B	S	6	12	18	24	30	Data Bits
	A	1	7	13	19	25	31	
	8	2	8	14	20	26	32	
	4	3	9	15	21	27	33	
	2	4	10	16	22	28	34	
	1	5	11	17	23	29	35	

Complete operational characteristics of both the IBM 7340 Hypertape Drive and IBM 7640 Hypertape Control are in the *IBM 7340 Hypertape Drive Reference Manual,* Form A22-6616.

The IBM 7340 Hypertape Drive and IBM 7640 Hypertape Control are attached to the 7090 System as shown in Figure 64. The 7909 Data Channel is a single-channel device, while the 7640 control is a two-channel device; therefore, simultaneous read-write operation requires that the 7640 be attached to two 7909 Data Channels. As many as ten 7340 drives may be attached to each channel of the 7640, making a possible total of 20 drives per 7640 control.

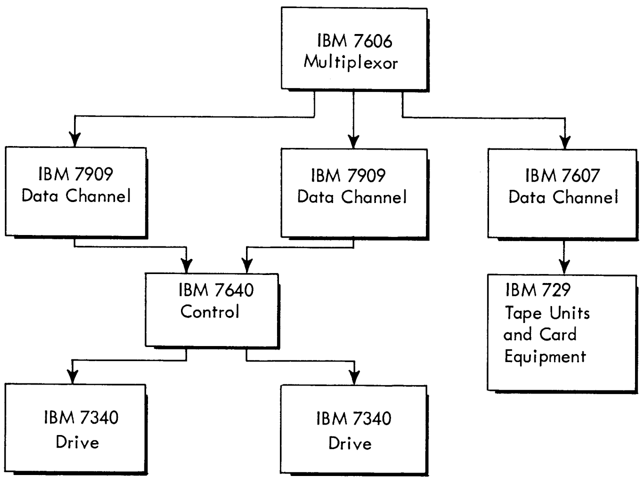


Figure 64. IBM 7090 System with IBM 7340 Hypertape Drive

Tape Motion and Markers

A single capstan moves the tape forward or backward at 112.5 inches per second. Rewinding occurs at 225 inches per second with tape in the vacuum columns.

Three markers appear on the tape to set off the recording area from the physical ends. About 25 feet from the physical beginning of tape is a marker called the beginning of tape (BOT). The BOT is adjacent to the machine edge of the tape.

About 25 feet from the physical end of tape is another marker called the end of tape (EOT). The EOT is the same size as the BOT but is located midway between the edges of tape. Tape motion stops immediately when the machine detects the EOT.

Because the 7340 cannot move tape forward past the EOT, a third marker, called the end warning area (EWA) marker precedes the EOT on tape by about 40 feet. The EWA marker is located adjacent to the edge of tape nearest the operator. When tape is being written, detection of the EWA does not stop tape motion but signals the end-of-tape condition.

Optional Features

Two optional features are available for use on the IBM 7909 Data Channel.

Read Backward Character Assembly and Storage facilitates processing of data received from a recorded tape being read in a backward direction by assembling the characters in reverse order.

BCD Translation Feature accomplishes binary-to-BCD character translation for magnetic tapes prepared or to be used by other IBM Data Processing Systems employing IBM 7340 Hypertape Drives.

Automatic Cartridge Loader Feature

The Automatic Cartridge Loader Feature may be used with the IBM 7340 Hypertape Drive (Figure 65). The automatic cartridge loader attaches to the top of the 7340 Hypertape Drive. The 7340 operator's control panel is repositioned to the top-front of the loader. The combined height of the two units approximates that of an IBM 729 Magnetic Tape Unit. With the loader installed, the two units act as one in all operations involving the loading and unloading of cartridges. The loader provides two additional cartridge positions for a total of four.

The four cartridge positions are:

Load Storage: A receiving station for all cartridges to be processed.

Discharge Storage: An eject station. The unload or change cartridge operation moves processed cartridges to this position. Processed cartridges are removed from this position by the operator.

Processing (or Loaded): The normal 7340 processing position. Reels are engaged in the drive hubs and tape is in the columns.

Unload (or Ready to Load): The unload or the ready to load position. Tape is out of the columns and the reels are disengaged from the drive hubs.

Operations

The cartridge loader automatically:

1. Unloads a cartridge from the unload position.
2. Loads a cartridge into the ready-to-load position.
3. Performs an unload-load sequence cycle. It unloads and moves a used cartridge to the discharge storage position, and takes the next cartridge from the load storage position and loads it into the processing position. The complete unload-load cycle time is about 30 seconds.

Operation of the Automatic Cartridge Loader Feature will be controlled by IBM programs supplied with the 7340 Hypertape Drive used with the IBM 7074, 7080, 7090, and 7094 Data Processing Systems.

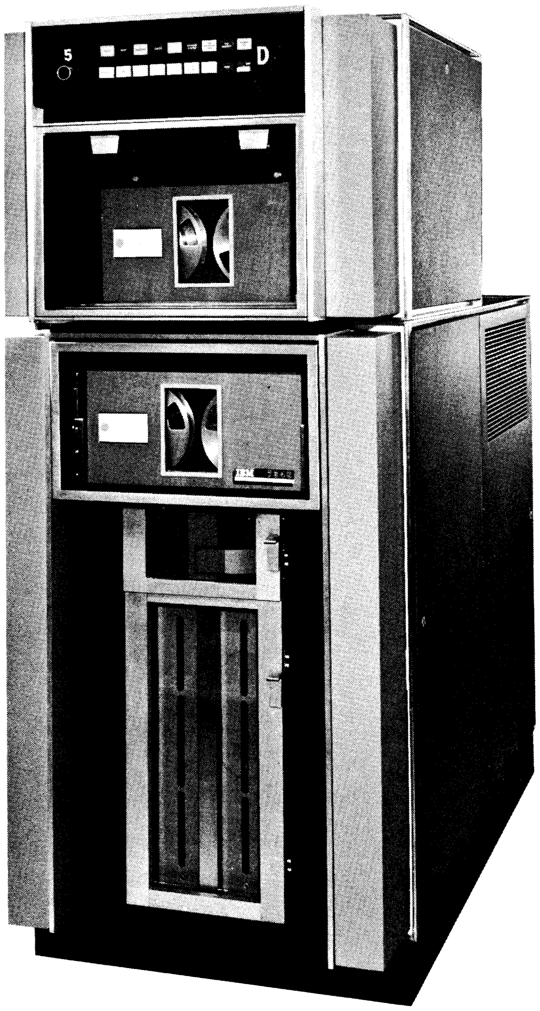


Figure 65. Automatic Cartridge Loader with IBM 7340 Hypertape Drive

A door in front of the loader provides access to the load storage and discharge storage position. The operator may open the door to remove or insert a cartridge while Hypertape is being processed, without interrupting processing. Opening the door during a change-cartridge operation stops the action, which is resumed when the door is closed.

Additional information is available in the *IBM 7340 Hypertape Drive Bulletin,* Form G22-6667.

IBM 1414 Model 6 Input-Output Synchronizer

The IBM 1414-6 Input-Output Synchronizer connects communication-oriented and paper tape devices to the 7090 System. Attachment to the 7090 is through the IBM 7909 Data Channel. Units that can be attached with the 1414-6 are (Figure 66):

IBM 1009 Data Transmission Unit
IBM 1011 Paper Tape Reader
IBM 1014 Remote Inquiry Units
Telegraph Input-Output Units

Six 80-character buffers, each assigned to a specific input-output device, are contained in the 1414-6. The buffers store up to 80 BCD characters and have an average data transfer rate of 11 microseconds per character to and from the 7909. The buffers are under 7090 program control and use normal interrupt procedures.

IBM 1009 Data Transmission Unit

The 7090 System, equipped with an IBM 1009 Data Transmission Unit, may function as a data processor and as a data transmitter or receiver. The 1009 unit allows not only two-way communication between two remote 7090 Systems, but also between a 7090 and an IBM 7701 or 7702 Magnetic Tape Transmission Terminal, an IBM 1013 Card Transmission Terminal, or another IBM system (1400-7000 Series) equipped with a 1009 Data Transmission Unit.

The 1009 Data Transmission Unit has four possible data rates: 75, 150, 250, or 300 characters per second. One 1009 may be attached to the 1414-6 input-output synchronizer and uses two of the six input-output buffers. A detailed description of the 1009 is contained in the *General Information Manual, IBM 1009 Data Transmission Unit,* Form D24-1039.

IBM 1011 Paper Tape Reader

The IBM 1011 Paper Tape Reader is an input device, controlled by the 7090 program in the same manner as other devices attached through the 7909 Data Channel. The reader operates at 500 paper tape characters per second, using either five-track telegraph or eight-track IBM tape. The tape can be chad or chadless and in the form of strips, reels, or rolls to be fed from the center.

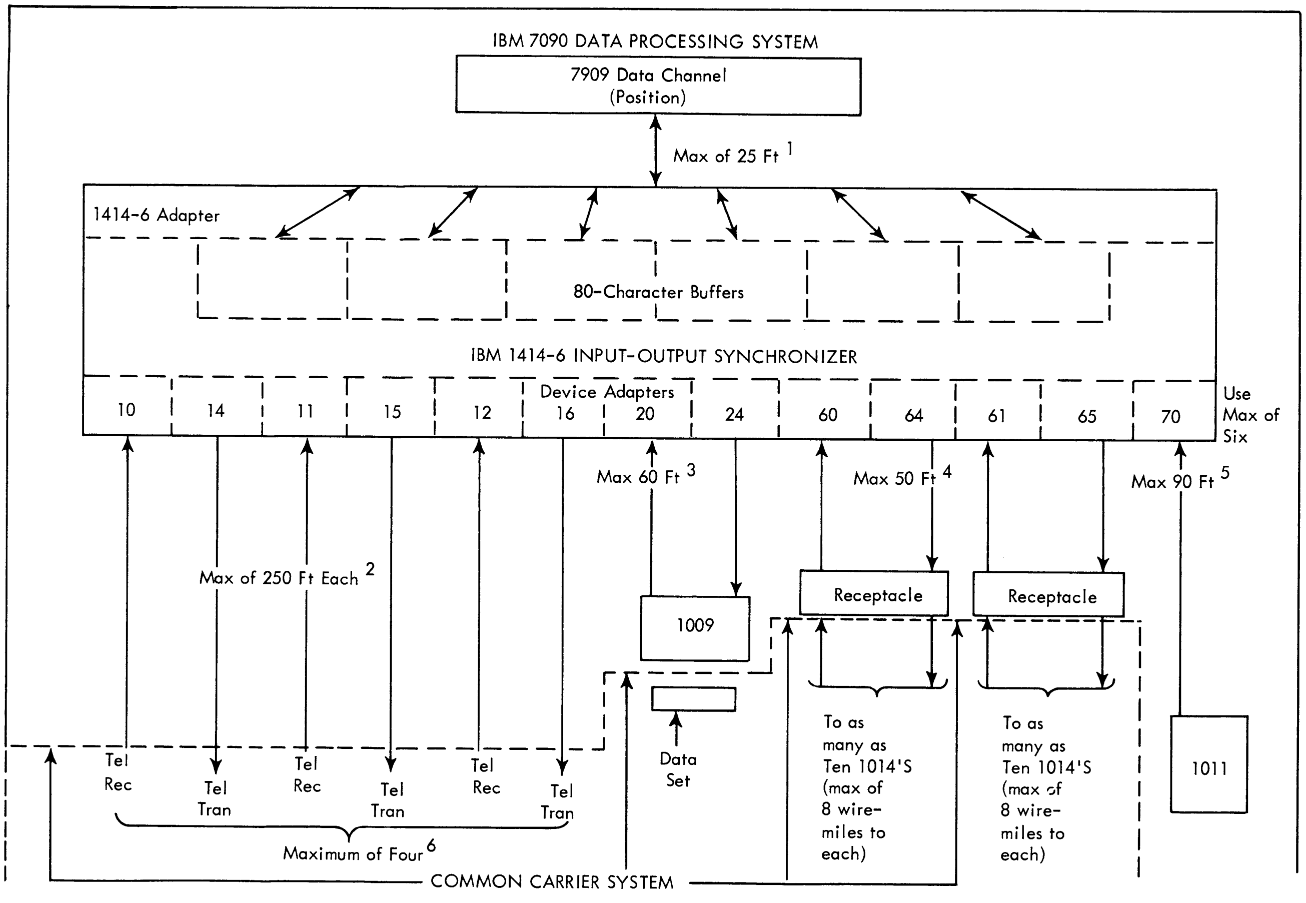
The 1011 Paper Tape Reader uses one of the six buffers for its operation. One 1011 reader may be attached to the 1414-6. A detailed description of the 1011 is contained in the *General Information Manual, IBM 1011 Paper Tape Reader,* Form D24-1044.

IBM 1014 Remote Inquiry Unit

The IBM 1014 Remote Inquiry Unit, with typewriter input and output, may be used as a means of system interrogation. The 1014 provides a visual record of information stored in or transmitted from the 7090 System. Remote inquiry provides direct access to any record stored within the 7090 System and furnishes a printed output under program control. The 1014 has a maximum data rate of 12½ characters per second for inquiry request and 15½ characters per second for inquiry reply, with as many as 78 characters per message on input-output operations. The first position of the buffer contains the identification of the 1014 being used. The position adjacent to the last inquiry character is a group mark (ǂ).

The 1414-6 may have one or two channels (adapters) for attachment of 1014 units. Each channel uses two of the six buffers, one for input and the other for output. As many as ten 1014 remote units are controlled by each adapter. However, only one 1014 per adapter may be in operation at one time. The address of the specific 1014 (0-9) is placed in the first position of the buffer.

The remote inquiry units are cable-connected to the 1414-6 and can be located 8 wire-miles away from



Notes
1. Data Transfer Rate--11 Microseconds/Character
2. Maximum Data Transfer Rate--Up to 10 Characters/Second
3. Possible Data Transfer Rates--75, 150, 250, 300 Characters/Second
4. Maximum Data Transfer Rates--12-1/2 Characters/Second (Inquiry Request)
   15-1/2 Characters/Second (Inquiry Reply
5. Data Transfer Rate--500 Characters/Second
6. One unit must be input and one unit must be output

Figure 66. Configuration of the IBM 1414-6 and Telecommunications Devices

the system. (A wire-mile is a parallel set of four wires, or two pairs of wires, one mile long.) For a detailed description of the remote inquiry unit, refer to the *"M" Bulletin, IBM 1014 Remote Inquiry Unit,* Form G24-1444.

Telegraph Input-Output

As many as four telegraph units[1] may be attached to the 1414-6 to communicate with remote input-output telegraph units. The data transmission rate of these connections can be up to approximately ten characters per second, depending on the transmission rate of the common carrier equipment used. The number of buffers used for telegraph communication equals the number of attached input and output units.

In receive operation, the telegraph units communicate with the 1414-6 by means of five-bit telegraph code. The 1414-6 translates this into standard BCD characters. The telegraph message is divided into two parts, the administrative portion (destination, sending station, date, time, and so on) and the data portion (body of the message). The data portion to be stored in the 7090 must be enclosed by parentheses. It should not exceed eighty BCD characters. With normal input-output programming, the possibility of the data channel being occupied with disk records or the possibility of combinations of other priority-sequenced operations makes it impossible to insure that subsequent buffer-loads of input characters will receive computer attention within the 100 milliseconds (time between characters) allotted for servicing the buffer after it is filled. Therefore, it is strongly recommended that data portions (to be stored) be limited to 80 characters.

The 1414-6 stores only one data portion in one buffer-load. Any additional parenthetical sections occurring within the message will be loaded in the buffer only if the previous data portion has been transferred to the 7909. If disk operations on the same channel or a combination of priorities have prevented the transfer, the new data are not received by the 1414-6.

Before the data go to the buffer, the 1414-6 automatically deletes the letters-shift, figures-shift, line-feed, and blank characters from the incoming message; these characters do not enter the buffer. Optionally, the carriage return and parentheses may be deleted.

In transmit operation, a reverse procedure is followed. Letters shift and figures shift are the only automatically inserted characters. The output message is brought from core storage to the buffer of the 1414-6, translated into telegraph code, and sent to the selected telegraph unit when the line is ready to receive it.

Addressing of Input-Output Devices

Each of the systems adapters attached to the 1414-6 is assigned a two-digit address to identify it to the 7090 program. The first digit identifies the type of adapter; the second digit is the number of that particular systems adapter and indicates the read-write status. Any combination of adapters shown in the following table may be attached to the 1414-6, provided that the combined buffer requirements do not exceed six. Possible input-output adapters, the assigned decimal address, and the number of buffers required for attachment are:

Adapter	Operation	Adapter Address	No. of Buffers Required
Telegraph	Read Write	10, 11, 12 14, 15, 16	2, 3, or 4
IBM 1009	Read Write	20 24	2
IBM 1014	Read Write	60, 61 64, 65	2 or 4
IBM 1011	Read	70	1
Any combination of the above adapters may be attached to the 1414-6, provided the combined buffer requirements do not exceed six.			

Address Register

The address register is a two-digit register that stores the address of the selected input or output device. The register is set as the result of a control operation (order) or a 1414-6 internal polling operation to designate a particular device requiring service. This requirement for service is called an *attention.* Attentions are caused by one or two conditions: (1) a write buffer becoming empty[2] (the empty buffer gives only one attention signal[3]), or (2) a read buffer being full. The register remains set until the input or output

1 One unit must be a transmitter and one a receiver; the other units may consist of a transmitter-receiver, one or two receivers, or one or two transmitters.
2 For 1009 write operation a delay occurs while accuracy of transmission is checked.
3 This attention is maintained until honored.

line is deselected (the address register is reset). Resetting the register results from:

1. End of read between the 7909 and the 1414-6 buffer (no error).

2. End of write between the 7909 and the 1414-6 buffer (no error).

3. End of sense between the 7909 and the 1414-6.

4. Start of control operation between the 7909 and the 1414-6.

If read-write operation is terminated with an UNUSUAL END signal, the register is not reset and the 1414-6 retains the status information that caused the UNUSUAL END. If the program does not interrogate the unusual condition at this time, the information is lost when the next command takes place. While the 1414-6 is retaining status information, attention requests will pile up, possibly leading to overrun conditions in which input information will be lost. A sense instruction should be given immediately after an attention or an UNUSUAL END to avoid this possibility.

For proper conditioning of the address register, these rules should be followed:

1. A read command or a write command must always be preceded by a control command.

2. A sense command must always be preceded by an UNUSUAL END, attention, or a control command.

3. An attention or UNUSUAL END should be immediately followed by a sense command to avoid losing real-time data that may be waiting for attention on another line. Also, be sure that the 7090 main program is written so that attentions may be honored immediately.

The control address uses the first digit to indicate the unit and the second digit to indicate the number of that type of unit and the read-write status. The 8 bit of the character is not used in the register. This character bit is not needed because the range of both unit and number is 0-7.

The sense command transfers four 4-bit bytes from the 1414-6 to the 7090. The sense data are transmitted to the 7090 over the A, 4, 2, and 1 bit lines. The following chart shows the assigned bit configuration for the status data:

Not Used	Program Check[1]	Data Check[2]	Exceptional Condition (Examine Byte 2)	Not Ready	Busy	Condition	No Transfer	Not Used	Tens Digit	Not Used	Units Digit
A	4	2	1	A	4	2	1	A	4 2 1	A	4 2 1
Byte 1				Byte 2				Byte 3		Byte 4	
Status								Address			

1. A program check is caused by selecting a write adapter followed by a read instruction or vice versa, or giving a sense command not preceded by: (1) a control command, or (2) either an unusual end or attention signal from the 1414-6.
2. A data check is caused by either: (1) a character parity check in the 1414-6, or (2) a 1414-6 machine check.

Because of the various units involved, the bit assignment of the second byte is general in nature and has a different meaning, depending on the unit and the type of operation performed. Figure 67 shows the bit error assignments for the different units and operations.

Adapter Number	Device	STATUS DATA INFORMATION			
		A bit - Not Ready	4 bit - Busy	2 bit - Condition	1 bit - No Transfer
1	Telegraph Read	Buffer not on line or power off	Buffer is being filled	Missed message	No request
1	Telegraph Write	Buffer not on line or power off	Buffer is being emptied	Last message in error; not transmitted to remote telegraph*	Last message transmitted but received incorrectly
2	IBM 1009 DTU Read	DTU not on line or power off	Buffer is being filled	Missed message	No request
2	IBM 1009 DTU Write	DTU not on line or power off	Buffer is being emptied	Last message in error; transmitted to local 1009, but not to remote 1009	Last message transmitted but received incorrectly
6	IBM 1014 Read	Buffer not on line or buffer power off	Not applicable	Not applicable	No request or buffer being filled
6	IBM 1014 Write	Buffer not on line or power off	Buffer being emptied	Last message in error; not transmitted	Last message not transmitted; station inoperative
7	IBM 1011 Read	Paper tape power off--out of tape	Buffer being filled	Not applicable	Not applicable

* May be transmitted to local telegraph

Figure 67. Detail Status Byte 2, Information

IBM 7607 Data Channels

A maximum of eight IBM 7607 Data Channels (Figure 68) may be used with the 7090 system.

As many as ten IBM 729 II or 729 IV Magnetic Tape Units, intermixed in any fashion, may be attached to each data channel in the 7090 system. Each channel may also have, in addition to the ten tape units, a card reader, card punch, and a printer attached to it.

Data Channel Select Registers

When a select instruction addressing a tape, card, or printer unit is interpreted in the CPU, it is sent to the specified data channel for execution.

If the select instruction is a data select (read select or write select) the instruction is placed in the channel's data select and unit registers. The operation to be performed and the type of unit involved is placed in the data select register. The unit number is placed in the unit register (Figure 69). Once the specified input-output unit has been selected, the unit register

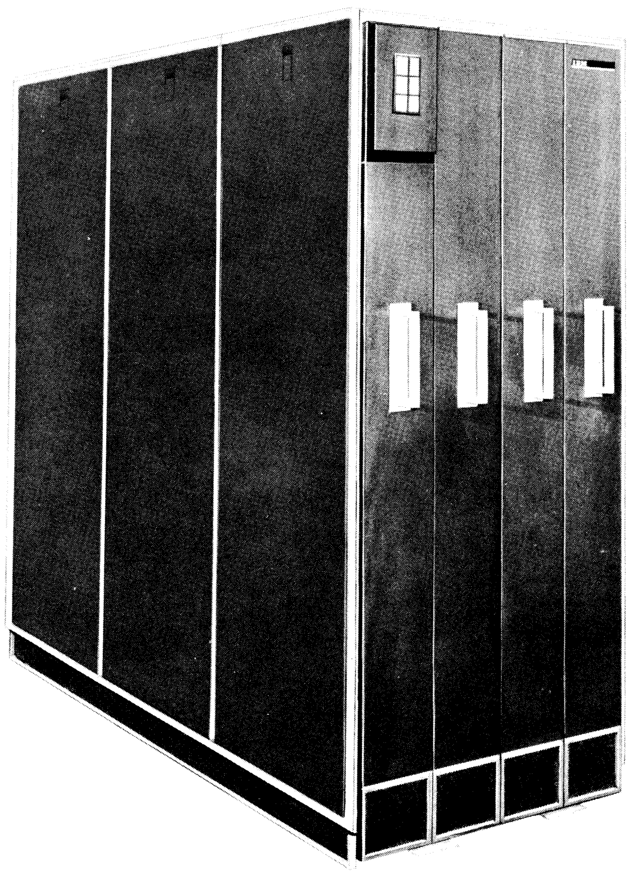
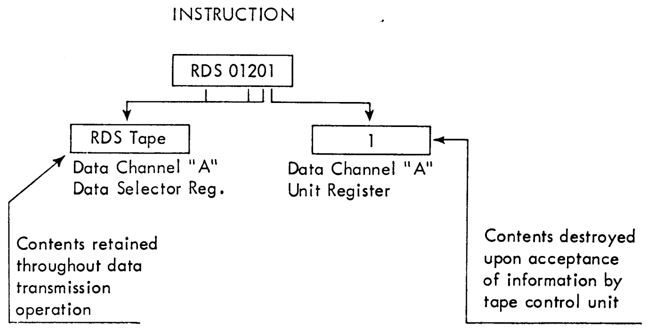


Figure 68. IBM 7607 Data Channel

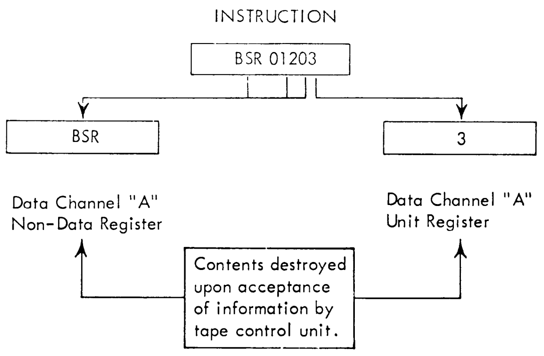


INSTRUCTION
RDS 01201
RDS Tape
Data Channel "A" Data Selector Reg.
1
Data Channel "A" Unit Register
Contents retained throughout data transmission operation
Contents destroyed upon acceptance of information by tape control unit

Figure 69. Data Channel Command Stacking

is free to accept new information sent from the main program. The data select register, however, is used throughout the entire input-output operation to control the data channel reading or writing activity. This register cannot accept another data select operation until the present operation has been terminated.

If the select instruction is a *non-data select* (backspace, rewind, or write-end-of-file instruction) the instruction is placed in the specified data channel's non-data and unit registers (Figure 70). The unit register has the same function as described for a data select. When the data channel executes a backspace record (BSR), backspace file (BSF), or a rewind (REW) instruction, the non-data and unit registers are used only until the tape units have been selected. This requires only a few microseconds. After this selection, the data channel is no longer required and is free to accept any select instruction sent from the CPU. In the case of a BSR or BSF, the execution is completed by the multiplexor. Any select instruction, addressing a card or printer unit and sent to the data channel before completion of a BSR or BSF, will be executed immediately.



INSTRUCTION
BSR 01203
BSR
3
Data Channel "A" Non-Data Register
Data Channel "A" Unit Register
Contents destroyed upon acceptance of information by tape control unit.

Figure 70. Data Channel Command Stacking

The REW operation differs from BSR and BSF in that it uses the tape control for a few milliseconds only. The operation is then controlled by the tape unit that is being rewound. Thus, once a REW has been started, any select instruction which does not address the tape unit being rewound will be executed immediately.

The WEF instruction is similar to a data select in that the specified data channel's non-data register is in use throughout its entire execution.

The data select, non-data, and unit registers of a channel are not directly addressable by the main program. The registers are of importance, however, in terms of synchronous timing relations between the main program and the data channel operation. For example, the outcome of the test made by a transfer-in-operation or transfer-not-in-operation instruction depends solely on the status of the channel's select registers.

Data Channel in Operation

The result of a transfer in operation or a transfer not in operation depends on the status of the channel's select registers. If either the data select or non-data register of a data channel is being used to hold information, the data channel is said to be *in operation.* If neither holds information, the channel is *not in operation.* Assume that a REW, BSR, or BSF addresses a channel not in operation whose attached tape unit is also not in use. When the execution of the REW, BSR, or BSF has been completed in the CPU and the computer proceeds to the next instruction, the channel remains in use for three machine cycles after which the non-data and unit registers are no longer in use. Then a transfer in operation given before the next select instruction will receive a negative response.

Data Channel Select Instruction Stacking

It is important in input-output programming, particularly with tape operations, to understand the instruction storing or "stacking" abilities of a data channel. Stacking reduces delays in the main program. The conditions described below can occur only when the multiple select instructions address either the same data channel, or a data channel and a magnetic drum.

The logical independence of each channel assures that the status of one channel is not affected (or does not affect) a select instruction addressing another channel.

When a select instruction is given in the main program and the specified data channel is not in operation, the select instruction is sent immediately to the channel and the main program continues without delay.

If a channel is in operation when a select is given in the main program, the data channel either: (1) does not permit the instruction to be sent, and the main program is delayed until the channel is ready to accept the instruction; or (2) accepts the instruction and stacks it in the select registers until it can be executed. In this case, the main program is not delayed.

Since data select instructions use the data select registers of the specified channel throughout the entire input-output operation, any data select given while another data select is in process will cause the main program to be delayed until the prior operation is completed.

If a data select addresses a card or printer unit, any non-data select given before its completion will delay the main program. This restriction is imposed so that the unit register may be used by the plus sense (PSE) instructions used with the punch and printer units.

Any select instruction given for approximately 50 milliseconds after a WEF operation has started will delay the main program. This implies that no select instruction can be stacked during this time. A WEF may be stacked, however, during the execution of another select instruction.

If a non-data select is given by the main program while the data select addressing a tape unit is in operation, the non-data select will be stacked. No delay will occur in the main program.

As previously indicated, a non-data select, with the exception of WEF, does not require the data channel once the tape unit has been selected. Therefore, any select instruction will be accepted by the channel if given during the execution of a non-data select. If the select instruction addresses a card machine or printer, it will be executed immediately. A select instruction addressing a tape unit will be stacked in the channel and any third select given during this time will delay the main program.

When a RDS or WRS is stacked in the channel, its associated reset and load channel instruction may be executed before it is. When this occurs, the channel command will be held until the input-output unit is ready for transmission. This condition does not disrupt the channel, the check indicator is not turned on and no delay results in the main program.

Programmed Channel Delay

In programming input-output activities, it is frequently desirable to synchronize the testing of the channel indicators with the I-O process. The load channel, store channel, transfer in operation, and transfer not in operation instructions provide flexibility in this type of programming.

A simple method which holds the main program at a given point consists of the following. At location Y, a transfer in operation is given the address of Y. Thus the main program will repeatedly execute the transfer until the specified data channel is logically disconnected. It should be noted that a channel is considered in operation as long as any of the select registers contain information. For example, consider the following sequence of instructions:

LOCATION	INSTRUCTION
100	RTDA 1201
101	RCHA 0600
102	BSRA 1203
103	TCOA 0103

The BSR instruction located in word 102 will be stacked in data channel A. The program will then delay at location 103 until tape unit 3 attached to data channel A has been selected.

Indefinite Delays

It is possible for a program to cause an indefinite delay in the system as a whole or in an attached data channel by selecting a unit not attached to the computer. If a select, reset-and-load, or load-channel instruction addresses a channel not attached to the computer or one with its automatic-manual switch set to MANUAL, the main program will delay indefinitely attempting to execute the instruction. If the channel is not attached, the select condition may be terminated by pressing the reset key on the console of the CPU. The computer may then be restarted. If the channel's automatic-manual switch is in manual status, when it is restored to automatic status a waiting select instruction will be executed and the program will resume. However, if the waiting instruction is a reset and load or a load channel, the input-output check indicator will be turned on when the channel is restored to automatic.

When (1) any select specifies an I-O unit which is not attached to the channel or is not in ready status, or (2) a WRS specifies a tape unit whose file protect ring has been removed, the select instruction is stacked and the channel will suspend operation indefinitely. The main program may proceed but will be delayed by the next select, or load channel instruction addressing the channel. If the specified I-O unit is not attached to the channel, the select condition may be terminated by pressing the reset key on the CPU console. If the I-O unit is not ready, appropriate manual service which readies the unit enables both the channel and the main program to continue operation. An exception is when the I-O unit is a tape that is not ready because it is rewinding. Upon completion of the rewind operation the tape unit is automatically restored to ready status and the program proceeds.

Figure 71 has been prepared to assist the programmer in determining when a channel is in operation.

Tape Check Indicator

When a transfer on redundancy check is used to test a write operation, it should be preceded by a channel delay instruction. This insures that, when the test is made, all bits including the longitudinal check bits of the last record written will be checked.

If a record or records are read by any combination of channel commands such that the operation is ended following an end-of-record recognition (an IORP command followed by an IOCD with a word count of zero), a similar situation prevails. A channel delay followed by a transfer on redundancy check provides a check on all bits including the longitudinal check bits of the last record read.

Select Operation	Channel In Operation Unit	Tape Unit In Use Until
RDS	Last storage reference has been made.	End of record
WRS	Longitudinal check character is read.	End of record
WEF	End-of-file gap and tape mark with its check character read by the read gap.	Same
BSR	Tape control selected.	Beginning of record gap + 25 ms.
BSF	Tape control selected	Beginning of file gap + 25 ms.
REW	Tape control selected	Load point.

Figure 71. Channel Delay Conditions

When word count control is used to govern the reading process (for example, an IOCD command with a word count of N) a channel delay followed by a transfer on redundancy check insures that the N words read the lateral and that *only* the lateral check bits have been checked. This is true if the record contains more than N words. When the channel word count register reaches zero, the channel has no way of knowing that it has read the last word of a record. The channel disconnect delays until one more transmission point has been reached or the LRCR has been read. However, tape errors occurring in subsequent information ($N <$ number of words in record), including the LRC, will turn on the channel tape check indicator.

End-of-File Indicator

When an end-of-file occurs following an RDS, the channel end-of-file indicator is turned on and the read operation is terminated immediately. Thus, if a channel delay is given at some point subsequent to an RDS and is followed by a transfer on end of file, the transfer condition is met if the channel has been disconnected by an EOF condition.

Such a test has validity only if this indicator is off at the time the RDS is executed by the channel, since this indicator may also be turned on either by other tape units or by a card reader.

Beginning-of-Tape Indicator

The beginning of tape indicator signals that a backspace operation has been completed by reaching load point rather than by sensing the beginning of a record (BSR) or an end-of-file record (BSF). The beginning-of-tape indicator may be turned on in the following ways:

1. When the tape is at its load point, a BSR or BSF instruction causes the indicator to be turned on immediately.

2. When the tape is positioned after any record in the first file: (1) two BSF instructions turn the indicator on if the file was preceded by an end-of-file record, or (2) one BSF instruction turns the indicator on if the file was not preceded by an end-of-file record. In case 1, the first BSF positions the tape over the tape mark in the end-of-file gap, and the second BSF returns the tape to its load point, turning on the beginning-of-tape indicator. In case 2, the first BSF moves the tape backward over the file to the load point, turning on the beginning-of-tape indicator.

3. When the tape is positioned after the first record of the first file: (1) three BSR instructions turn the indicator on if the record was preceded by an end-of-file record, or (2) two BSR instructions turn the indicator on if the record was not preceded by an end-of-file record. In case 1, the first BSR moves the tape to the beginning of the first record, the second BSR moves the tape to the beginning of the end-of-file record, and the third BSR moves the tape to its load point, turning on the beginning-of-tape indicator.

To test whether a specific backspace instruction has turned the indicator on, the instruction should be followed by a data or non-data instruction and a channel delay instruction (such as TCOA) before the BTT instruction is used. Once the indicator is on, it will remain on until turned off by the execution of the BTT instruction or a manual reset. The BTT instruction should not be used if more than one tape is being backspaced on the same channel.

End-of-Tape Indicator

The end-of-tape indicator can be turned on only during the execution of a WRS or WEF instruction. The specified channel remains in operation throughout the execution of either of these instructions until the tape unit is disconnected. Therefore, the combination of a WRS or WEF instruction, followed by a channel delay, followed by an ETT instruction determines whether or not the end-of-tape marker has been passed at any time prior to the writing of the terminal end-of-record gap.

If the physical end-of-tape marker is encountered while the tape unit is being stopped, the channel's end-of-tape indicator will be turned on but will not be recognized until a succeeding ETT instruction is executed. This situation can occur when the end-of-tape marker falls in an end-of-record gap.

End-of-File Sensing

The WEF instruction writes an end-of-file gap and a tape mark (including its check character). End-of-file gaps recorded with the WEF instruction are not checked for noise, but both the tape mark and its longitudinal check characters are read and both are checked laterally and also longitudinally.

The recognition of an end of file occurs with the execution of a RDS instruction when the end-of-file is spaced over. Assuming that records and an end-of-file have been written on a tape, a program to sense end of file is shown in the example in Figure 72.

ADDRESS	INSTRUCTION		COMMENTS
00100	RDS	01202	Select tape 1, channel A
00101	RCHA	00500	Load the IOCD command
00102	TCOA	00102	Channel delay
00103	TEFA	01000	Transfer on EOF
00104	HTR		
00500	IOCD	77777 0 05000	Start reading into location 5000
01000	REWA	01201	Rewind tape 1
01001	TR		Transfer out

Figure 72. Program Example

With the program shown in the program example, tape records are read into consecutive storage locations beginning with location 5000. When the EOF is sensed, the channel is disconnected and the program executes the instruction at 00102. This instruction loops until the channel is not in operation; then the instruction to test for end-of-file is executed.

Blank Tape Sections

If, at the beginning of a record, no data are provided to be written in that record, a blank section of approximately 3¾ inches will be written. When the blank section has been written (end of record) the channel will either proceed, transfer, trap, or disconnect according to the present command.

Examples are:

1. A WRS instruction not followed by an RCH instruction (or followed by an RCH which loads an IOCD with a word count that is initially zero).
2. An IORP or IORT command with word count initially zero that is loaded at the beginning of a record.
3. A WRS, followed by an RCH referring to an IOCT or IOST command with a word count of zero. If command trap is enabled, the trap will occur after the blank tape section has been written and the channel has left operation.

The blank sections of tape are always read and checked by the read gap of the tape unit. Any noise that is present in the section will turn the tape check indicator on. The channel will remain in operation in all cases until such checking has been accomplished.

To maintain compatibility with 704-type tapes, care should be taken when information is rewritten several times, beginning at some fixed point (other than load point) on tape. The sequence of WRS and BSR instructions may not be repeated more than ten times. If this limit is exceeded, the record gap preceding the record being rewritten may be lengthened. However, if the record preceding the one being rewritten is always reread by the execution of two BSR's and an RDS before the rewriting, no such restrictions apply. A file may be rewritten from a fixed point (other than load point) only once if the sequence of BSF, WEF, and WRS is used. A WEF following the BSF will result in a longer end-of-file gap. If an RDS is used to space over the EOF gap instead of the WEF, the same restrictions do not apply. If the last record of the preceding file is read prior to the rewriting of the file, no such restrictions apply.

A tape may be rewritten in its entirety or from a fixed point forward, but new records and files may not be inserted between existing records and files.

Programmed Interruption of a Data Channel

It may be desirable during the course of a calculation to alter or stop an I-O activity. If a reset and load channel is given while an I-O operation is in progress, a new command will be loaded into the channel immediately, regardless of the possible loss of data. This feature may be used to change the sequence of or to interrupt an I-O operation.

A reset and load channel which loads an IOCD, IOCT, or IOST command with a zero word count can be used to terminate an I-O operation in progress. For example, a read operation is started by the instructions in location 200 and 201. (Figure 73 shows all addresses and locations in the octal system.) Fifty words are to be read from tape unit 1 into core storage beginning with location 800. Assuming that the read operation will not have been completed when the reset and load channel instruction at location 215 is executed, the disconnect command (location 501) will be loaded into the channel and the I-O operation will be stopped immediately.

LOCATION	INSTRUCTION		COMMENTS
200	RDS	1221	Read-select tape 1 Channel "A"
201	RCHA	0500	Load command into Channel "A"
.	.	.	.
.	.	.	.
215	RCHA	0501	Load command to interrupt Channel "A"
.	.	.	
.	.	.	.
500	IOCD 00062 0 01440		
501	IOCD 00000 0 00000		

Figure 73. Programmed Interruption

If a data select instruction is being stacked when a disconnect command is being loaded, the channel is selected, and the execution of the disconnect command is delayed. If a non-data select instruction is being stacked and a disconnect command is loaded, the channel will remain in operation until the non-data select instruction is executed.

If a store-channel is given prior to the execution of an interrupting reset-and-load-channel, a word may be transmitted between core storage and the I-O device before the execution of the reset-and-load-channel. This is true even if the two instructions are given consecutively.

Data Channel Timing

Once a channel initiates a storage reference cycle, it will continue to take such cycles until all of its requirements are met. A channel will require one cycle for: (1) each data word transmitted to or from storage, (2) each additional command loaded into it, and (3) indirect addressing of a 7090 command.

When a channel has taken the number of cycles required, a test is automatically made to determine if any other channels require reference cycles. If so, each channel will take, in turn, the number of cycles each one needs.

The requirements for all channels in operation must be met within a period of 24 cycles on the 7090. If this number is exceeded, one or more of the channels will be disconnected and the I-O check indicator will be turned on.

EXAMPLE: Channels A and B are to be in operation at the same time. Channel A is controlled by the sequence of commands IOCP, TCH, and IOCD. Channel B is controlled by the sequence of commands IORP, TCH, TCH and IOCD. Thus, the maximum number of consecutive cycles that may occur is seven (three for channel A—one data-transmission cycle, one command cycle for the TCH, and one command cycle for the IOCD—and four for channel B—one data transmission cycle, two command cycles for the TCH's, and one command cycle for the IOCD).

The maximum allowable program time (in microseconds) between successive load channel instructions for a channel using tape is given by the formula:

$T_{LC} = 2.18C\ (29 - M) - (I + 6.6)$ on a 7090, where:

C = word count of the command loaded by the load channel.

M = number of consecutive cycles that may be taken by all channels in operation.

I = 2.18 if the load channel is indirectly addressed, 0 if not.

EXAMPLE: Channel C is in operation and load-channel instructions are to be used. The commands to be loaded by the load-channel have a word count of two. Further, channels A and B are operating as described in the above example. Thus, the maximum time which may be safely used by the main program between successive load-channels is:

$T_{LC} = 2.18 \times 2\ (29 - 8) - (0 + 6.6) = 83.8$ microseconds

A data channel relinquishes priority between the time the last word is transmitted by an IOCP or IOSP command and the arrival of a subsequent channel command. A store channel instruction may, therefore, store an address which is one greater than the address of the last word transmitted by the IOCP or IOSP command.

Magnetic Tape Timing

Since magnetic tape involves physical motion and mechanical drives, the variations of tape speeds and distances are large when contrasted with speeds in the internal computer.

It is recommended, therefore, that programs use the transfer-in-operation, transfer-not-in-operation, load-channel, and store-channel instructions in synchronizing I-O activities rather than depending on the timing associated with physical tape motion.

With respect to timing, select instructions may be regarded as subject to execution at four different levels:

1. Initially, the main program executes the select instruction by sending it to the channel. If the channel is not in operation, the main program execution requires two cycles.
2. The channel is used by a data select instruction throughout the operation and until the tape control disconnects the tape. The channel is used by a non-data select instruction for stacking purposes only. If the channel is free initially, a non-data select instruction is executed in the channel. The channel remains in use for three cycles.
3. Except for a rewind operation, the tape control is in use for most, and in some cases all, of the time that the tape is in motion.
4. The time during which the tape itself is in motion is either identical to or exceeds the time in which the tape control is in use.

Timing, Reading and Writing

Minimum and maximum times for tape operation differ for writing and reading. During writing, time variations are related only to the duration of the recording pulses. The speed of the tape itself affects only the distances between lateral rows as they are recorded on the tape. During reading, however, variations in the tape drive on the reading unit as well as those which occurred when the tape was written will affect the time of transmission.

If the transfer-in-operation, the transfer-not-in-operation, reset-and-load-channel, load-channel, and store-channel are not used to synchronize I-O activities, the following formulas are given for the purpose of calculating the approximate tape passing time of tape files (time in milliseconds):

729 II and V (200 cpi)	T = .4N + 10.8R + 53F
729 II and V (556 cpi)	.15N + 10.8R + 53F
729 IV and VI (200 cpi)	.27N + 7.3R + 36F
729 IV and VI (556 cpi)	.1N + 7.3R + 36F
729 V (800 cpi)	.1N + 10.8R + 53F
729 VI (800 cpi)	.07N + 7.3R + 36F

Where: N = total number of words in all records.

R = number of records involved.

F = number of files or file gaps involved.

Note that if the tape is positioned at its load point when the operation is started, the beginning-of-tape gap should be counted as an extra file gap in computing the value F.

Since a channel is always disconnected when an end of file is sensed and may be disconnected between records, T may represent the sum of several distinct time intervals. During each such interval, both the channel and the tape control, to which the tape is attached, are in operation. The total time during which a tape is in motion may or may not exceed T, depending upon the main program.

Timing, Tape Check, and End-of-File Indicators

If N in the formula for T is exactly equal to the number of words in one or more records, the longitudinal check bits of the last record will have been read in a time not exceeding T. Discrepancies occurring in or before the longitudinal check bits will have turned on the channel's tape check indicator by T time. Similarly, if N and R represent exactly the number of words and records preceding an end-of-file gap, then the expression for T with $F = 1$ ($F = 2$ if the tape was initially at its load point) yields the time that may elapse before the channel's end-of-file indicator is turned on.

Timing, Reset and Load Channel

Once the execution of a WRS or RDS instruction is begun in a channel, a reset-and-load-channel must supply the channel with a command before the tape is ready to send or receive the first data word. The minimum time within which the instruction must be given depends on whether a WRS or RDS is specified and the position of the tape when selected. Time is in milliseconds (ms).

INSTRUCTION STARTING TAPE MOTION	TAPE POSITION WHEN INSTRUCTION IS EXECUTED	TIME WITHIN WHICH RESET-AND-LOAD-CHANNEL MUST BE GIVEN 729 II, V	729 IV, VI
WRS	Not at load point	4 ms.	2.7 ms.
WRS	At load point	30 ms.	20 ms.
RDS	Not at load point	1 ms.	.7 ms.
RDS	At load point	15 ms.	10 ms.

The read gap is positioned $3/10$ inch behind the write gap and must read the longitudinal check bits before the channel can disconnect and receive a select for the next record. During reading, the first word of the record is sent to the channel and then to a storage location as soon as the channel can make a storage reference cycle. During writing, however, the first word to be written is taken from storage at the time the reset-and-load-channel is executed. It is held in the channel until the tape is positioned to write the first character of the word.

Timing for BSR, BSF, and REW

The following table may be used to assist in calculating running time involving backward movement of tape. The time required to backspace the tape over a record, file, or to rewind the tape to its load point may be computed by adding:

1. Time to start tape moving in a backward direction,
2. Time to space the tape, and
3. Time to stop the tape.

The times, which may depend on the status of the tape, are given below and are average times:

$N = .402 \times$ number of words	(729 II and V – 200 cpi)
$.144 \times$ number of words	(729 II and V – 556 cpi)
$.264 \times$ number of words	(729 IV and VI – 200 cpi)
$.096 \times$ number of words	(729 IV and VI – 556 cpi)
$.102 \times$ number of words	(729 V – 800 cpi)
$.066 \times$ number of words	(729 VI – 800 cpi)

$R = 10.8 \times$ number of records (729 II)
$= 7.3 \times$ number of records (729 IV)

$F = 40 \times$ number of files plus 1 (729 II)
$= 34 \times$ number of files plus 1 (729 IV)

Note that $F + 1$ must be used for rewind to account for the beginning-of-file gap. If a rewind is given for a tape positioned more than approximately 450 feet from its load point, then about 1.2 minutes are used by the high-speed rewind operation. For a tape positioned at its load point, none of the backward moving select instructions cause tape motion. However, they may use the tape control from 25 to 3,000 microseconds in testing the load point condition. Consequently, a following tape select instruction using the same channel may be subject to a slight delay.

Keeping a Tape in Motion

Once started, the tape moves at a constant speed until a channel command causes a disconnect or an EOF is encountered. The tape continues its motion to the middle of the end-of-record gap and then stops. If a WRS or RDS for the next record has been issued by the main program and is awaiting execution when the tape is ready to stop, the new select instruction will be in effect and the tape motion will continue at full speed.

Failure to keep a tape in motion will not increase the total running time of the program.

If the tape receives a new select instruction while it is slowing, acceleration occurs again. The tape drive is designed so that regardless of the tape motion when it begins acceleration, the time taken to reach the beginning of the next record is constant.

Similarly, when a tape is to stop, the time taken to move from the end of the last record to the point on the tape where the tape control disconnects the tape is constant (approximately 2.5 and 4.25 ms. for the read and write gaps, respectively).

Card Reader

One IBM 711 Card Reader (Figure 74) may be attached to any channel on the system.

The 711 reads cards at a rate of 250 cards per minute. Cards to be read are placed in the feed hopper face down, 9-edge first. Information punched in the cards may be decimal, alphabetic, binary, or any special character code. The reading format is controlled by the stored program and a control panel located on the reader. With a 711 Reader attached to a channel, a 716 Printer must also be attached to the same channel because power is supplied to both the reader and punch from the printer.

The reading of cards is started by the execution ot an RDS instruction addressing a channel and an attached reader. Physical motion in the reader is started and within 55 ms. (following the RDS), a reset and load channel instruction must supply the channel with its first command. Any sequence of commands calling for the uninterrupted transmission of at least 24 words causes the reading of the entire card. The words are transmitted in the order: 9-row left, 9-row right through 12-row right. For example, if an IOCD command with a word count of 24 is initially executed, the 24 words are read from the card in the indicated order and stored in 24 consecutive core storage locations beginning at the address specified by the command. After execution of the IOCD command, the channel and the reader will be disconnected. With a word count of less than 24, only the specified number of words is read into core storage, the channel and the

Figure 74. IBM 711 Card Reader

reader are disconnected, and the remaining words are spaced over without being read. With a word count greater than 24, the channel reads from successive cards until the word count is zero. Words on a card may be read into non-consecutive locations through use of an IOCP, IOCT or other count control commands which do not necessarily disconnect when the word count reaches zero.

Commands specify record control functions exactly as they do for tape records. Thus, an IORP command with a word count of 100 reads the entire card (24 words) after which an end-of-record occurs in the reader and the channel proceeds to its next command in sequence. For an IORP command with a word count of 1, the channel reads the 9-row-left word, spaces over the remaining 23 words, and, as the card reader end-of-record occurs, proceeds to the next command in sequence.

When the hopper of the reader becomes empty, operation is suspended in the reader and channel. The channel read-write register is in use throughout the delay period. The main program is delayed if another select instruction addressing the channel in question requires execution. If additional cards are placed in the hopper and the reader start key is pressed, normal reading will continue. If no additional cards are placed in the hopper and the operator presses the reader start key, the remaining cards which have not been read (last two cards) are read in a normal manner. If the card reader and channel are selected after the last card is read, an end-of-file indicator in the channel is turned on and both devices are disconnected.

Card Reader Timing

NOTE: In the following card machine descriptions, all times shown on the timing circles are minimum times with single-channel operation. Minimum time is defined as the maximum allowable time between consecutive load channel instructions and includes the time necessary for the execution of the load channel instruction.

In continuous reading, cards are processed at a rate of 250 per minute. The timing of a card cycle is shown in Figure 75.

If the reader is disconnected between two cycles, the new RDS for the second cycle must be given in the hatched portion of the cycle to insure continued reading at full speed. The reader speed is constant if the new select is given within 30 ms. following the reading of the 12-row right. If the select is given between 30 and 90 ms., a delay of 60 ms. is imposed before reselection occurs. Card reader speed will be reduced to 200 cards per minute if this is done continuously.

When the select is given in the hatched portion of the cycle, a reset and load channel must supply the channel with its first command within 85 ms. following the 12-row right transmission. This provides for safe timing, as the average elapsed time between the 12-row right for one card and the 9-row left for the next is 108 ms.

When the card reader is not in motion and the first RDS is executed, the position of the reader with respect to its cycle is arbitrary. Hence a reset and load channel must be executed within 55 ms. although the 9-row left transmission will not occur, on the average, for 110 ms.

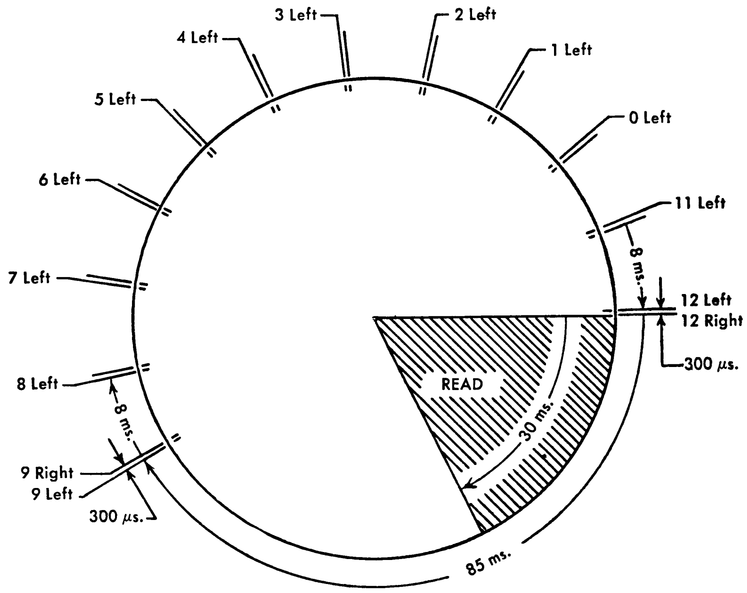


Figure 75. Card Reader Timing Circle

The times at which successive left and right words are transmitted to storage are shown in Figure 75 by the designations 9-left, 9-right, and so on. The minimum times between word transmissions (300 μs. between left and right words and 8 ms. between the right word of one row and the left word of the next row) should be considered when load channel instructions are used to synchronize the CPU and the reader. For example, assume that a reset and load channel has supplied the DSC with an IOCT command whose word count is one. This command sends the 9-left word from the reader, through the channel, to the location in core storage specified by the address field of the IOCT command. The word count is reduced to zero when the word enters storage. At this time a load channel instruction must be waiting in the main program. Assume that another IOCT command with a word count of one is loaded in the channel, and that such a command is sent to the channel at every transmission point. Then the times between transmission (alternately 300 μs. and 8 ms.) would represent the maximum allowable times between consecutive load channel instructions.

When the word count of an IOCD command is reduced to zero, the reader is disconnected on what would have been the next transmission point. A reset and load channel which loads an IOCD, IOCT, or IOST command with a zero word count may be used to terminate an I-O operation. If the RCHA is given within 125 microseconds after any read-right transmission point, or 925 microseconds after an end-of-record point, the card reader will immediately disconnect. If RCHA is given later, the disconnect occurs at the next transmission or end-of-record point (even if this next point occurs on the following card).

Timing Chart

When the reader is operating at normal speed each cycle requires 240 ms. The cycle is divided into 360°. One degree corresponds to an average time of ⅔ ms. The card reader cycle is also divided into 20 cycle points (18° for each cycle point). The beginning or end of most impulses coincides with a cycle point. The relation between the timing chart (Figure 76) and the timing circle shown in Figure 75 is as follows:

The 9 row of the card passes under the reading brushes at 9°. Whenever holes are punched in the 9 row of the card, impulses which last until 18° are available at the corresponding read brush hubs. With any 72 read brush hubs wired to the 72 calc entry left and right hubs, these impulses are transmitted in the form of two 36-bit words, to the channel and then to some core storage locations. Transmission points occur between two and three ms. after the reading brush falls into the hole punched in the card. (The contents of the word register in the channel are also reduced at this time.) Thus, the 9-left and 9-right transmission points occur at 12° or 13°. Each read brush impulse has a duration of 9°.

The end of the record occurs at 224° and the end-of-file indicator is turned on, after the last card is read, at 7° of the following cycle.

When the reader is disconnected it continues to move through its cycle until 330° (asterisk on the timing chart); at this point the feed unit is latched or locked in position. The driving mechanism continues to turn, however, and when a new RDS is executed the mechanism turns until it reaches one of four possible unlatching positions. At this point a new card reader cycle begins. The beginning point is always 330° because of an interlock.

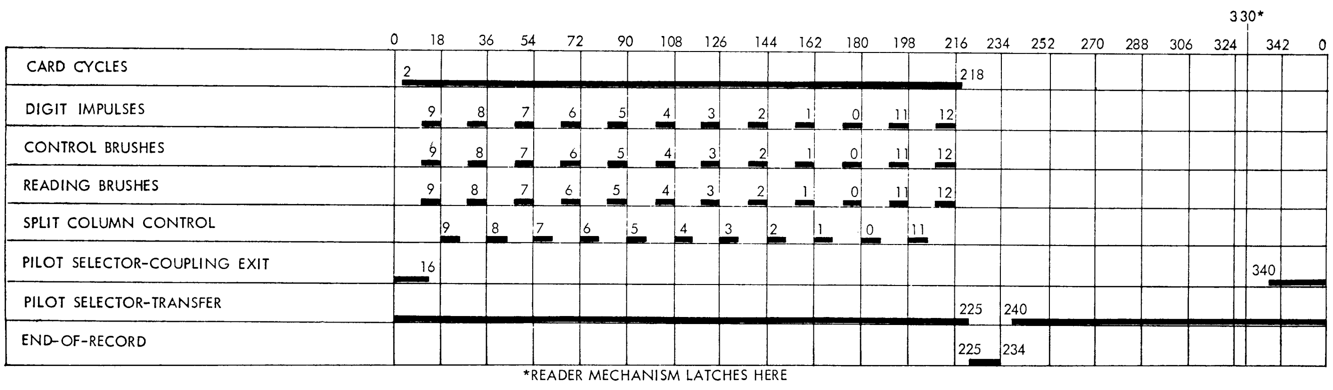


Figure 76. Card Reader Timing Chart

Card Punch

One IBM 721 Card Punch (Figure 77) may be attached to any channel on the system.

Punched card output may be decimal, alphabetic, binary, or any special character code. Information is punched at the rate of 100 cards per minute. Cards to be punched are placed in the punch feed hopper face down, 9-edge first. The punching format is controlled both by the stored program and a control panel located on the punch unit. Data channels having a punch attached must also have a 716 printer attached.

Basic operations are analogous to those of the reader, except that instead of building up a card image in core storage with information read from a card, the channel sends a card image from core storage to the punch to be recorded on a card.

Punching is started with the execution of a WRS whose address specifies a channel and attached punch. Physical motion is started in the punch by the WRS execution, and within 70 ms. an RCHA in the main program must supply the channel with its first command. The punch will be disconnected if this does not occur; then when the reset and load channel is executed (if no other channel component is selected) it will turn on the I-O check indicator on the computer console. Although any sequence of commands might follow as a channel program, assume that only one order (an IOCD with a word count of 24) is executed. Starting with the location specified by the IOCD command, 24 words from consecutive locations in storage are punched as the 9-row left, 9-row right, through the 12-row right of the card. The specified card columns punched are controlled by the control panel wiring; the 36 calc exit left and the 36 calc exit right hubs may be wired to any 72 of the 80 punch magnet hubs. Each of the 72 bit positions in a pair of left and right words of the card image then correspond to a particular column on the card. Thus a 1 in position S of the third word transmitted causes an impulse to appear at the calc exit left S hub. If this hub is wired to punch magnet position 1, a punched hole in row 8, column 1 of the card results.

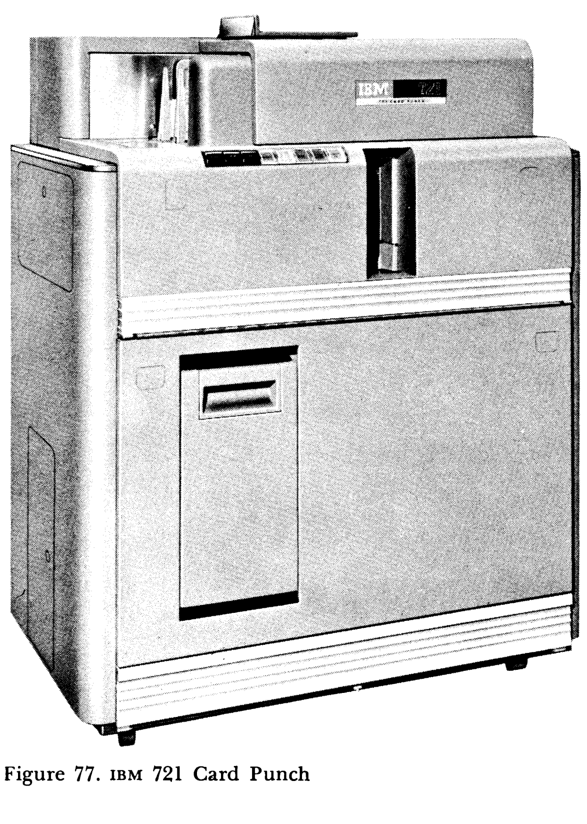


Figure 77. IBM 721 Card Punch

Punch Timing

During continuous punching, cards are processed at a 100 card-per-minute rate. A punch cycle is illustrated in the timing circle shown in Figure 78.

If an IOCD command causes a punch to disconnect at any time during or after the punching of a card, a new WRS must be executed within 25 ms. following transmission of the 12-right word to the punch (before the end of the hatched portion) if the 100 card-per-minute rate is to be continued. If this is done an RCHA must supply the channel with its first command for the new cycle within 95 ms. following the 12-right word transmission to the punch for the previous cycle. If the punch is disconnected and a new WRS is executed after the end of the hatched portion of the cycle, the punch may have mechanically latched or stopped. In this case punch motion has ceased except for the drive mechanism, which coasts to a stop. Motion is resumed when the new select is executed, but approximately one revolution may be required before the drive reaches a point where it will unlatch the punch.

When the punch is at rest and the first WRS is executed, the drive is positioned arbitrarily. An RCHA must always supply the channel with its first command within 70 ms. If the RCHA is not given soon enough, the punch and channel may be disconnected, in which case the RCHA will turn on the I-O check indicator. The minimum time between word transmission (300 μs. between words and 31 ms. between rows) should be considered when load channel instructions are used to synchronize the punch with the CPU.

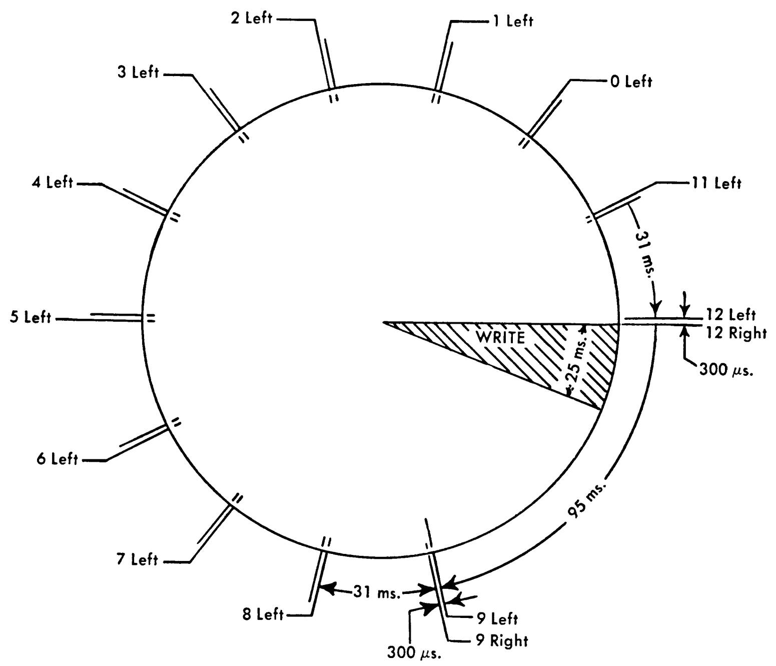


Figure 78. Card Punch Timing Circle

Timing — Channel Commands

It is important to note that the points designated as 9-left, 9-right, and so on, in Figure 78 represent times at which card image words are sent from the channel to the card punch. Using load or store channel instructions, these times may be taken as cycle points that afford the main program the means of identifying which phase of a punch cycle is in process at any given time. Note the following properties of the channel which may be relevant in this regard:

1. As soon as a word is sent to the punch, the next word to be transmitted is sent to the channel. The channel address and word count registers are increased and decreased, respectively, by one. Thus the *N*th point on the timing circle represents the time:
   a. the *N*th word is sent to the card punch.
   b. the *N*-plus-1 word is sent to the channel.
   c. the contents of the address register are increased to the *N*-plus-2 word address.

When an RCHA is executed, the first command and the first data word specified by the command are sent to the channel. The contents of the word register and address register are stepped down and up by one. These actions in the channel are all started by the RCHA execution, and all take place before the next instruction is executed in the main program. The foregoing account applies only to the punch, but is applicable to other I-O units during write operation.

2. The loading of the first command and first data word when a reset and load channel follows a WRS is such that, once a cycle has been taken for a storage reference, the channel continues to take successive cycles until all of its logical needs have been met. For example, when a TCH command is received by the channel, the next command (address of the TCH) is taken from core storage during the next storage cycle. Furthermore, all needs for storage cycles in all channels must be satisfied before the main program is permitted to make another storage reference.

3. An exception to the channel's tendency to take new data or command words at the earliest possible cycle is the load-channel instruction. This exception is illustrated by an IOCT command given during writing. In accordance with item 1 above, the word count will be reduced to zero when the last word has been sent from core storage to the channel, but before the last word has been sent to the punch. The channel does not look for a waiting load-channel instruction in the main program until the word count is zero and the last word has been sent to the punch.

4. When an IOCD punch command is executed, the punch is disconnected and the channel drops out of operation when the word count plus one transmission point has been reached.

Items 3 and 4 may be summarized as follows: An IOCT command with a word count equal to *N* words forces the channel to look for a waiting load channel instruction in the main program at the *N*th transmission point (Figure 78). An IOCD command with a word count of *N* words results in a normal disconnect at the $N + 1$ transmission point.

5. An end-of-record occurs approximately at the end of the hatched portion of the cycle, no sooner than 20 ms. and an average of 25 ms. following the transmission of the 12-right word to the punch. The end-of-record condition in a punch, occurring at the 25th transmission point, is similar to an end-of-record gap on tape. Unlike tape, card equipment has no means of writing an end-of-file.

A reset and load channel instruction which loads an IOCD, IOCT, or IOST command with a zero word count may be used to terminate an I-O operation. If the RCHA is given within 925 microseconds after any transmission or end-of-record point, the card punch will immediately disconnect. If RCHA is given later, the disconnect occurs at the next transmission or end-of-record point (even if this next point occurs on the following card).

Timing Chart

A simplified timing chart of the punch is shown in Figure 79. The basic cycle of 600 ms. is broken into 14 subsections, divided by cycle points numbered 12, 13, 14, 9 . . 12. Cycle points 9 through 12 denote the times at which pairs of words are sent from the channel to the punch for normal row-at-a-time punching. More exactly, points 9 through 12 stand for the time at which the left word of the row is sent to the punch, with the right word following in about 300 μs. (Figure 78.

Cycle point 13 represents the time at which an end-of-record condition occurs. Notice that cycle point numbers correspond to card row numbers. Numbers like 14.5 or 9.5 are to be taken as points standing halfway between 14 and 9 or between 9 and 8. The fractional part of a number always implies an extension to the right of the cycle point. This left-to-right orientation may also be thought of as the direction in which the cycle actually progresses in time. The point D (denoted by *) indicates the starting point of the cycle. The timing chart shown in Figure 79 should not be used as a mechanical reference. The card punch is effectively locked at point D whenever it is not in motion.

Impulses shown in the timing chart are:

Card Cycles. Each feed cycle, an impulse is available at these hubs.

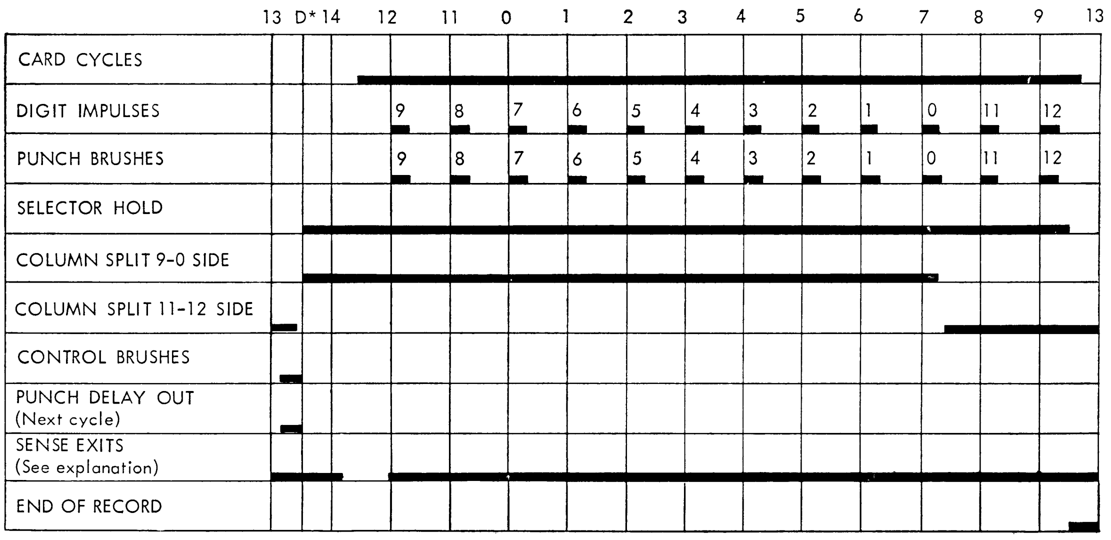
Digit Impulse. These impulses are available each feed cycle.

Punch Brushes. These impulses are available when the card is read by the punch brushes. A card passes over the punch brushes and is read during the cycle following the one in which it was punched.

Control Punches. Control punches in the 8 row of a card are sensed and produce impulses at these hubs on a punch cycle. Six such control brushes are available and may be manually set to read any six of 80 card columns.

Punch Delay Out. An impulse is available on the punch cycle following the cycle in which a control punch is sensed if the control brush is wired to a punch control IN hub.

Column Split. The column split acts as an internally wired selector that transfers from the 9-0 side to the 11-12 side after 0.4 time on the timing chart. The contacts return to their normal side at 13.4.



* PUNCH MECHANISM LATCHES HERE

Figure 79. Card Punch Timing Chart

Selector Hold. The 10-position selectors are of the immediate pickup type. Whenever a selector pickup hub is impulsed, its contacts transfer and remain transferred until 12.5 of the punch cycle, at which time the common hubs are again connected to the normal hubs. If, however, at 12.5 the pickup hubs are still impulsed (by a sense exit, for example), the selector remains transferred until 12.5 of some succeeding punch cycle.

Sense Exits 1 and 2. Impulses are made available at these hubs through the execution of appropriately addressed PSE instructions in the main program. In general, synchronous relation between the CPU and the punch need not concern the programmer unless PSE instructions are to be used to pick up selectors or exert some control over punching operations. Points of interest regarding the execution of a PSE which addresses a sense exit hub on the punch are:

1. When addressing the punch, a PSE must be given while the channel is in operation and the punch is selected; i.e., the order must be executed after an initial WRS in the main program and before a disconnect occurs in the channel. Otherwise, the PSE has no effect and is treated as a no-operation.
2. When the punch is not in motion and the first WRS is executed, the feed is effectively locked at "D" (Figure 79). The cycle does not start until the arbitrarily positioned drive mechanism has moved to the latch point and the feed becomes unlatched (able to move). If a PSE is given immediately following the initial WRS, the impulse is available before the punch cycle has started.
3. Under the circumstances indicated in item 2, the impulse is emitted from the sense exit hub from the time the PSE is executed to 14.2, and from 12.6 through 14.2 of every cycle thereafter, until the punch is disconnected. Once the punch is disconnected, the impulse is not available until another identically addressed PSE is executed. Thus, if an IORP command is followed by an IOCD with a zero word count, the punch is disconnected at 12.6 (end-of-record) and the sense exit impulse is no longer available. If a WRS is executed immediately after the disconnect, punching continues without losing the next cycle. Notice that if it is desirable to use sense exit impulses during some cycles but not during others, some scheme such as that described in the preceding explanation is necessary to terminate the hub's impulse without terminating continuous punching.
4. A PSE instruction executed between 14.2 and 14.9 causes an exit impulse to be emitted at the end of that cycle and on all succeeding cycles from 12.6 to 14.2.
5. A PSE instruction, given any time at cycle point 9 or later, causes an impulse to be emitted from the appropriate sense exit within 6 ms. and lasting until 14.2. The impulse will be repeated in the usual manner until the punch is disconnected. On the last cycle of punch operation a PSE should not be programmed after 9 time. Once the impulse is emitted it remains active until 14.2 of the next cycle, even if the punch is disconnected. Internal damage may occur if the emitting impulse is present at the sense exit hub throughout a period when the punch is not in use.

Printer

One IBM 716 Printer (Figure 80) may be attached to any channel on the system.

The printer is equipped with 120 rotary type wheels (Figure 81). Each wheel has 48 characters including numerals, alphabetic symbols and special characters (Figure 82). Use of the stored program enables the computer to print any desired information in any form convenient to the programmer. This information is printed at the rate of 150 lines per minute. Printing format is controlled by the arrangement of the information in storage and by a control panel located on the printer.

Figure 80. IBM 716 Printer

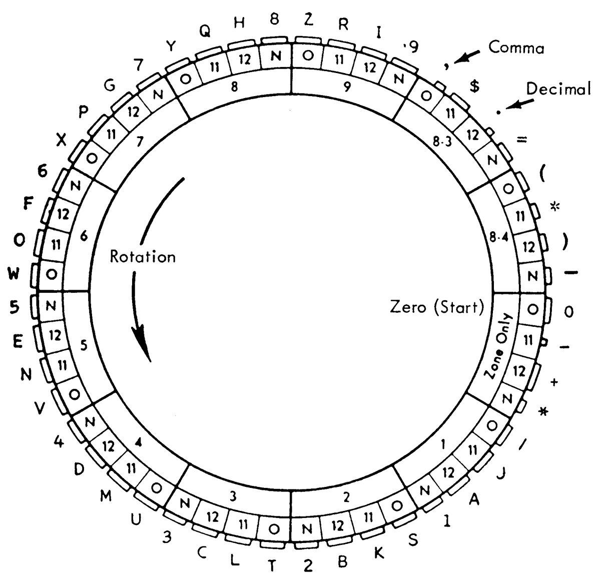


Figure 81. Type Wheel Schematic (FORTRAN)

digit	no (N) zone	12 (Y) zone	11 (X) zone	0 zone
no digit		+	–	0
1	1	A	J	/
2	2	B	K	S
3	3	C	L	T
4	4	D	M	U
5	5	E	N	V
6	6	F	O	W
7	7	G	P	X
8	8	H	Q	Y
9	9	I	R	Z
8-3	=	.	$	,
8-4	–	)	*	(

Figure 83. Punched Card Code (FORTRAN)

A line is printed in a time period which is called a print cycle. During this interval the type wheel is rotated until the specified character is centered in front of the platen and then printed. The amount of rotation, and consequently the character printed, depends on the time in the print cycle when the electrical impulse initiates motion. For example, if a print wheel receives an electrical impulse during that part of the cycle designated as 9 time, the number 9 is printed. Also if the print wheel receives an impulse at 1 time and an impulse at 12 time, the print unit positions the wheel to print the letter A (according to the standard IBM code shown in Figure 83).

Printing is similar to tape and card operations with respect to the role played by the channel to which the printer is attached. A printer record corresponds to a single printed line or to the information in core storage necessary to print one line. An end-of-record condition occurs at the end of each print cycle. Consequently, all eight channel commands are available for use in a print program.

Printing a Line

To initiate printing, a WRS with a normal address specifying a channel and its attached printer may be employed. Physical motion in the printer is caused by the WRS and within 58 ms. a reset and load channel must be executed to supply the channel with its first command.

If an IOCD command with a word count of 24 is given, the channel synchronizes itself with the printer and sends 24 words from consecutive locations in core storage, each at an appropriate time, to the printer; thus one line will be printed. The characters which are printed are determined by the contents of the 24 words in core storage.

The first two words are sent to the printer at 9-time in the printer cycle. If both words contained all 1's (S, 1-35) and all other words in the card image contained 0's, the number 9 would be printed in 72 positions on the same line. The printer control panel has 72 hubs (calc exit left and right) which correspond to the 72 bit positions of the pair of words transmitted from core storage. Wherever a bit position in the first word contains a 1, an impulse is emitted by the corresponding calc exit left hub; likewise a bit in the second word creates an impulse at a calc exit right hub. The 72 hubs may be wired directly to any 72 of the 120 print entry hubs. Impulses sent to the printer via these hubs control the type wheels directly and cause specific characters to be printed. Thus, through wiring between the calc exit and the print entry hubs, each of the 72 bits from a pair of words in core storage may be made to correspond to a particular type wheel.

Zone Digit	12 None	12 8-3	12 8-4	11 None	11 8-3	11 8-4	0 8-3	0 8-4	0 1	None 8-3	None 8-4
A	&	.	⌑	–	$	*	,	%	/	#	@
B	/	.	⌑	–	$	*	,	%	&	#	@
C	&	.	⌑	–	$	*	,	%	0	#	@
D	–	.	⌑	–	$	*	,	%	/	#	@
E					(This code not used)						
F	+	.	)	–	$	*	,	(	/	=	–
G	+	.	⌑	–	$	*	,	%	/	+	–

Figure 82. Alternate Type Wheel Characters

The print cycle, as it progresses, goes through points designated 9-time, 8-time, . . . 0-time, 11-time, 12-time. These times are analogous to those of the standard 407 that operates from card reading. At each time point, the next pair of words from a 24-word record in core storage are sent to the printer. The 24 words are designated 9-left, 9-right, and so on, corresponding to the time points at which they are sent to the printer. The record comprises a card image of exactly the type obtained by reading a card into consecutive core storage locations with each pair of words corresponding to a card row.

Note that the character printed by a given type wheel is determined by the contents of a fixed bit position in every other word of the record. Thus, if the 1-left and 12-left words have one bits and all other left words have zeros, the type wheels associated with the bit positions will print the letter A.

Printing Multiple Lines

If the word count of an IOCD command is greater than 24, additional lines may be printed. The 25th word is taken as the first word of the card image for the second line of printing; i.e., the 25th and 26th words are sent to the printer at 9-time of the second print cycle and successive words will be sent to the printer and printing will continue until the word count reaches zero. When the count reaches zero, the printer is disconnected from the channel and, if no other operation is waiting, the channel drops out of operation. If this occurs in the middle of a card image, the line is printed as though all of the missing portion of the image contained zeros.

In general any command or sequence of commands may be used in sending words to the printer. An end-of-record condition occurs following the transmission of every 24 words.

Printing with Checking

The previous statements describe printing without checking. Checking is possible because the printer can not only receive print impulses from the computer but can also send back "echo impulses" generated by the individual type-wheel position. Printing with checking requires a somewhat more complicated program, but can be accomplished without reducing the 150-line-per minute printing speed. Via control panel wiring, echo impulses may be returned to core storage in word groups which are similar to the words that were sent to the printer to be printed. The stored program can then compare the image transmitted against the image received, to insure that the type wheels were correctly positioned. During the first half of the print cycle words are sent to the printer, and during the last half of the cycle the echo words are returned from the printer. These two time periods overlap somewhat, so that echo words begin returning from the printer before all print words have been sent. The general sequence of events in a print cycle with echo checking is as follows:

1. An RDS instruction addressing a channel and its attached printer is given to initiate the first print cycle.

2. Within 58 ms. a reset and load channel must be executed, thus supplying the channel with its first command.

3. As in printing without checking, 12 pairs of words (forming the 12 rows of the card image) are taken by the channel from storage and are sent to the printer, where they determine which characters are printed. Nine pairs of echo words, similar to rows 9 through 1 of the original image, are sent from the printer back to core storage. In the standard IBM code, the digits 8-3 and 8-4 (with or without zone punching) are used for special characters. Two additional pairs of echo words provide a check that the correct print wheels have received these impulses. As a result, for all positions in the original image where both 8 and 3 rows (or 8 and 4 rows) contain 1's, the 8-3 (or 8-4) pair of echo words will contain 1's. A single type wheel causes the emission of only one echo whenever it prints a character. Zone (0-11-12) printing is not echo checked. If such characters requiring both a zone and a digit impulse are printed, only the digit impulse may be echo checked.

In printing with echo checking, an end-of-record condition occurs after 46 words have been transmitted (24 words from storage and 22 echo words returned). The exact sequence of transmission is: 9-left through 1-right written; 8-4 left and right echo words received; 0-left and right words written; 8-3 left and right echos received; 11-left and right words written; 9-left and right echos received; 12-left and right words are written; 8-left through 1-right echos received.

It is possible for the main program to compute throughout the cycle. If the channel is operated through an IOCD command with a word count of 46, it is necessary to stretch the image out over a 46-word block of storage, reserving appropriate locations for the returning echo words. By using a sequence of commands, it is possible to print an image of 24 words and also direct the echo words to any other configuration of storage locations.

Printing More than 72 Characters per Line

Through use of selectors or column splits on the printer control panel, storage can activate more than 72 print wheels. For example, seven 10-digit numbers with signs can be printed. Additional characters can also be printed by impulses emitted by the printer itself. The control panel may also be wired so that 120 characters originating from storage can be printed on each line at the rate of 75 lines per minute, with two print cycles being required for each line of printing. Normal spacing of the printer must be suppressed before the second cycle.

Timing without Echo Checking

Continuous printing occurs at a rate of 150 lines per minute, each line requiring one normal print cycle. The timing of such a cycle is shown in Figure 84. If the transmission of the 12-right word causes the channel to drop out of operation, the next WRS must be given within 115 ms. following the 12-right transmission, to keep the printer in continuous motion. The reset and load channel which follows the select must be given within 173 ms. of the 12-right transmission to supply the channel with its first command for the second print cycle. These minimum times allow for the safe timing, as the average elapsed time between the 12-right transmission of one print cycle and the 9-left transmission of the next cycle is 216 ms.

In general, it is not necessary to give a new select for each print cycle. Any desired sequence of commands may be used to print lines continuously and an IOCD command will terminate printing when it is executed by the channel. The channel to which the printer is attached remains in operation during the printing operation. The time between print cycles is sufficient to permit the execution of some other I-O operations through the same channel with no interruption in the 150 line speed. When another I-O operation is used between the printing of two lines, it is necessary to disconnect the channel following the last word transmission to the printer and to issue a new WRS during the hatched portion of the cycle for printing of the next line.

Printer Disconnect

An IOCD with a word count of 24 disconnects the printer on end-of-record. An IOCP with a word count of 24 and an IOCD with a word count of zero disconnects on the 12-row right. An IOCT with a word count of 24 and without a load channel waiting also disconnects on the 12-row right. As the disconnect occurs, the channel is taken out of operation. Any select instruction (including the WRS for the next print cycle) issued while the channel is still in operation delays the main program until the printer disconnects and frees the channel to accept a new select. An IOCD command may be used to disconnect the printer before 24 words have been transmitted. For this operation the channel drops out of operation one cycle point after the last word is sent to the printer. When the channel drops out of operation, the printer itself remains in motion. It is for this reason that a new WRS issued at any point in the hatched portion of the cycle enables printing to continue at maximum

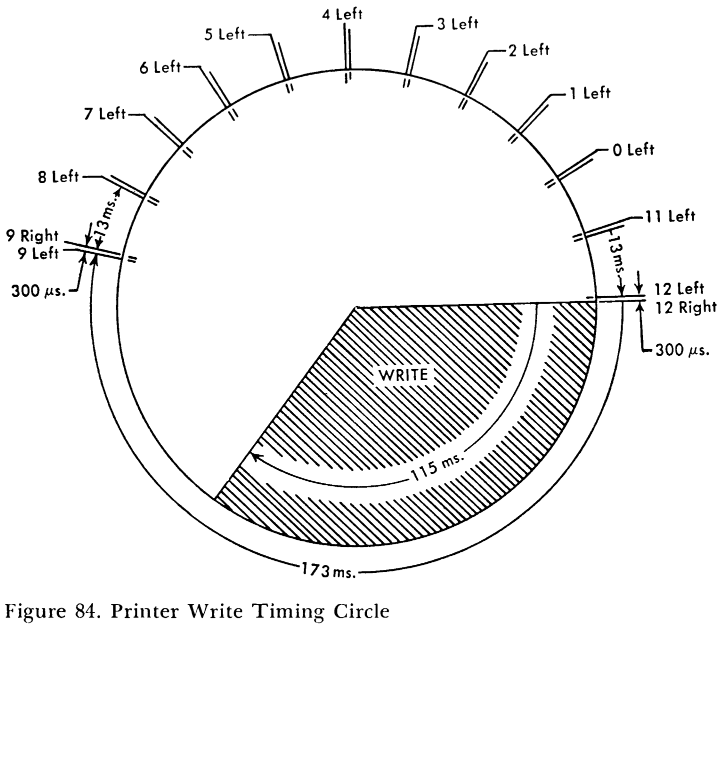


Figure 84. Printer Write Timing Circle

speed. A new WRS, given more than 115 ms. after the 12-right transmission, may find the print unit latched. In this case printer motion has ceased except for the drive shaft (which coasts to a stop). The shaft will resume full speed when the new select is executed but will require about one revolution in which to return to a point where it can latch with the print unit again. Thus, the effect of giving a WRS too late is to lose one print cycle. This affects printing speed and may affect control panel wiring, but no logical interference with the main program or the channel is entailed.

A similar timing situation is involved when the printer is not in motion and the first WRS is executed. The drive shaft will be at an arbitrary position. If the shaft is positioned just before the latch point, the print cycle may be started within a few milliseconds. In terms of the cycle shown in Figure 84, this corresponds to a WRS given at the end of the hatched portion. Because of this possibility, a reset and load channel must always be given within 58 ms. following the first WRS. The average elapsed time between the first WRS execution and the 9-left transmission to the printer is 280 ms. If a reset and load channel is not given soon enough, the printer and the channel may be disconnected. A late reset and load turns on the I-O check indicator.

Other minimum times shown in Figure 84 should be considered when using load channel instructions to synchronize the main program and the printer. For example, assume that a reset and load channel loads an IOCT with a word count of one. This command sends the 9-left word to the channel. At 9-left time this word goes to the printer. A load channel must be waiting in the main program. If the waiting load instruction then loads the printer channel with the same command, a second load channel must be given within 300 microseconds. If the same IOCT command is loaded for each word transmitted in the image, then a total of 23 load channels are required.

Timing of Data Channel Commands

It is important to note that all points on the timing circle of Figure 84 represent times at which card image words are sent from the channel to the printer. Other events which come into play at this time are:

1. When one card image word is sent to the printer, the next word is sent to the channel. The channel address register and the word count register are increased and decreased, respectively, by one. Thus, the Nth point on the timing circle represents the time at which the Nth word is sent to the printer, the $N+1$ word is sent to the channel, and the address register is increased to the address of the $N+2$ word.

2. A general principle of channel design is that, once a cycle has been taken for a storage reference, the channel continues to take storage cycles until all of its needs are met. For example, during a write operation, when an IOCP command is executed and the last data word enters the data register, the contents of the word register go to zero. At the next transmission point the contents of the data register are sent to the printer. At this point a storage cycle is started to bring the next command from storage and is followed by another storage cycle to bring in the next data word.

During a read operation, when an IOCP command is executed and the last data word enters the data register, a storage cycle is taken to store the contents of the data register. During this cycle the contents of the word register go to zero, thus starting another storage cycle to bring in the next command. Similarly, when the TCH command is received by a channel, the command specified by its address field is immediately loaded into the channel (followed by the first data word). Furthermore, if more than one channel is in need of storage references at one time, then all such references will be made.

3. If synchronous operation with the main program is involved, action in the channel is not always immediate. During a write operation, as indicated in item 1, when the word count has been reduced to zero, the last data word (subject to the current command) has been stored in the channel but has not been sent to the I-O device. With an IOCT command being executed after the last word has been written, a waiting load channel will then be executed. With one reset and load channel and 23 load channels, both conditions (word register equal to zero and the data word sent to the I-O unit) are met at each transmission point before a new load channel can be executed in the main program.

4. Similarly, if an IOCD command is executed, the printer is disconnected and the channel drops out of operation when the transmission point of word count plus one has been reached.

A reset and load channel instruction which loads an IOCD, IOCT or IOST command with a zero word count may be used to terminate an I-O operation. If the RCHA is given within 925 microseconds after any transmission or end-of-record point, the printer will immediately disconnect. With RCHA given later, the disconnect occurs at the next transmission or end-of-record point (even if this next point occurs on the following print cycle).

5. An end-of-record condition in a print cycle occurs no sooner than 13 ms. after the 12-right trans-

mission. At this time any end-of-record response specified by the current command is made.

Although the instruction WRS has comparable meanings for both magnetic tape and printers (words are sent from storage to the I-O device) the execution of commands of the record control type differs during writing, depending on whether a tape or printer is selected. During a print cycle, an end-of-record condition always occurs at a fixed point of the cycle. If the command (write end-of-record) with a word count of 2 is given to the printer, this will result in printing the 9 row. The remainder of the print image will be skipped over and the next command will not be brought into the channel until the end-of-record occurs.

Write Binary

The address of a WRS printer instruction may specify binary printing. After such a select has been issued, no words are transmitted to the printer until 1-time. Two words, corresponding to the 1-left and 1-right rows, are sent to the printer. This results in the printing of 1's by the type wheels, corresponding to the non-zero bit positions of the two words. On each cycle, only these two words are sent to the printer, so 0's can be printed by control panel wiring only. The transmission of such words during binary printing is accomplished through execution of channel commands in the usual way. The end-of-record occurs the same as with the write printer operation.

Timing with Echo Checking

As shown in Figure 85, the timing of a print cycle with echo checking is similar to that of one without it. Following the first RDS, an RCHA must supply the channel with its first command within 58 ms. The time point at which the 24 card image words are transmitted to the printer are unchanged by use of echo checking. With echo checking, however, 22 additional transmission points occur to provide for the echo words. If the printer is disconnected after the last transmission (1-right echo), a new RDS must be given within a minimum of 12 ms. (hatched portion) if printer motion is to be continuous. When an IORP or IORT command is used, the channel remains in operation and approximately 16.3 ms. (and no sooner than 12 ms. after 1-right echo) an end-of-record occurs. In general any number of printed lines may be echo checked following the execution of a single RDS. An RDS addressing a channel and attached printer represents the only instance in which a select instruction initiates both writing and reading.

If an IOCT command with a word count of N is the first command executed in a print cycle, a waiting load channel is executed at the Nth transmission point, whether this specifies the transmission of an echo word or a print image word.

NOTE: Position 19 of the command, indicating non-transmitting mode, is effective only on read printer operation and provides a useful method for skipping the 8-4 and 8-3 rows of numerical printing (assuming that these rows are not used for sign printing).

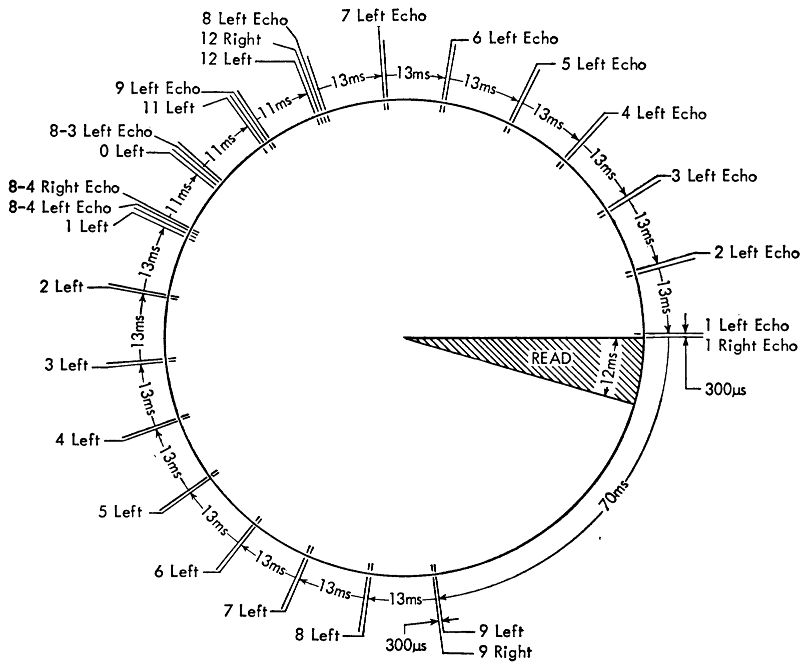


Figure 85. Printer Read Timing Circle

Programming Examples

A computer program is similar to the program received at baseball games, concerts, and many other presentations in that it is a plan of operations or events that will occur. The process of getting to work each morning may be compared to a program concerned with the following problem: Compute A + B − C and store the result (D), if it is a plus number; if minus, halt the computer (Figure 86).

Block diagrams, also called flow charts, are a schematic diagram of the logic of the computer and methods it uses in solving a problem. The main reason for a flow chart is that it is easier to write and understand than a written paragraph about the problem. The flow chart is a map of all logic paths and decisions used by the computer, and simplifies the writing of a coded computer program.

The same program used in Figure 86 may be expressed in program terminology as shown in Figure 87. Given: Factor *A* stored in location 100, factor *B* in 200, factor *C* in 300.

The instruction location designates the place, in core storage, where the instruction is stored. The instruction abbreviations are such that they represent the actual operation involved. For example, SUB means subtract and STO means store, while CLA means clear the register to zero and add. The address part designates a location in core storage where a number is located or where a number may be stored. Thus, the operation of the program would proceed as follows.

The program is started with the first instruction (CLA 100) which is contained in location 0000. This instruction will clear the accumulator register to zero and then bring the contents of core location 100 into the accumulator (factor *A*). The next instruction (ADD 200) will bring factor *B* from storage and combine it with factor *A*. The third instruction (SUB 300) brings factor *C* from storage and subtracts it from the combined factors *A* and *B*. The fourth instruction then takes the result in the accumulator and stores it in storage location 400. Thus *D* has been formed and stored.

Instruction Location	Instruction	Address
Move A 0000	CLA	100
Form A + B 0001	ADD	200
Form A + B − C 0002	SUB	300
Form Answer (D) 0003	STO	400

Figure 87. Simple Program

A possible use of two of the shifting instructions is shown in Figure 88 with the following facts known. Two numbers are contained in the same storage location. One number is located in positions 6 through 20, and the other is in positions 21 through 35. (Assume that this word is already located in the accumulator.) The problem is to multiply the number in positions 6-20 by the number in positions 21-35.

Location	Instruction	Address	Comments
0000	LRS	0015	Move positions 21-35 into the MQ.
0001	RQL	0016	Align this number in proper place to be used as the multiplier.
0002	STO	0100	Store the multiplicand so that it may be used in the multiplication.
0003	MPY	0100	Multiply the two numbers.
0004	STQ	0200	Store the result. (STQ is used because the result is small enough to be completely contained in MQ.)

Figure 88. Multiply and Shifting Problem

Conditional transfers may be used to solve the following type of problem. Assume that *A* and *B* are two positive numbers located in storage at locations 100

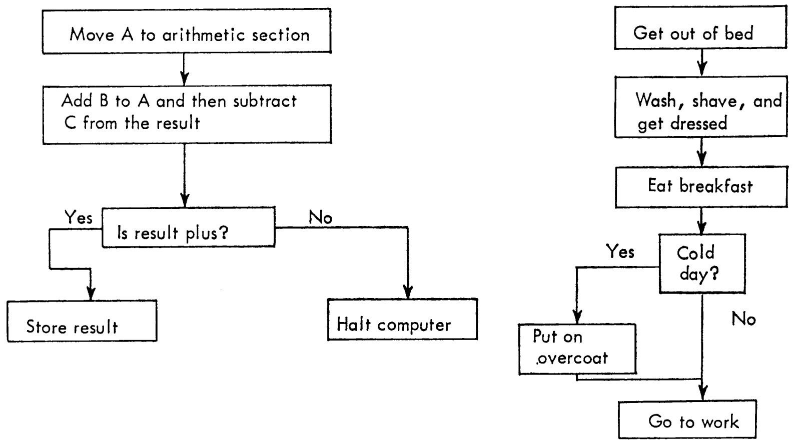


Figure 86. Simple Program Analogy

Location	Instruction	Address	Remarks
0000	CLA	0100	Factor A
0001	SUB	0101	Subtract factor B from factor A
0002	TZE	0017	Factors are equal
0003	TPL	0005	A is larger than B
0004	TMI	0012	A is smaller than B
0005	CLA	0100	Factor A
0006	STO	0201	Store A
0007	CLA	0101	Factor B
0010	STO	0200	Store B
0011	HTR	0022	Stop. A was larger than B
0012	CLA	0100	Factor A
0013	STO	0200	
0014	CLA	0101	Factor B
0015	STO	0201	
0016	HTR	0022	Stop. A was smaller than B
0017	CLA	0100	Factor A
0020	STO	0200	
0021	HTR	0022	Stop. A was equal to B
0022	Proceed with program		

Figure 89. Flow Chart and Program for Sorting

and 101. The problem is to find the smaller number and put it in location 200; also, to place the larger number in 201. If they are equal, put one number in 200 and nothing in 201. The computer program is shown in Figure 89.

The use of index registers can be pointed up by showing the number of program steps saved, and thus also computer time saved. Given numerical constants in locations 1 through 50, with a 1 in location 100, the numerical value 50 in location 200 and the value 50 stored in location 300. The problem is to add 1 to each of the 50 constants. Figure 90 shows the problem solved without using index registers. Figure 91 shows the same problem solved with index registers being used. The advantages and flexibility of indexing are readily evident.

Comment	Location	Instruction	Address
Form constant plus	1000	CLA	0001
one and	1001	ADD	0100
store	1002	STO	0001
Increase constant	1003	CLA	1000
address and	1004	ADD	0100
store	1005	STA	1000
	1006	CLA	1002
	1007	ADD	0100
	1010	STA	1002
Reduce the counter	1011	CLA	0300
by one	1012	SUB	0100
Test	1013	TNZ	1000
Stop	1014	HLT	

Figure 90. Address Modification without Indexing

	Location	Instruction		Address
Set 50 in XRA	1000	LXA	A	0200
Constant modifica-	1001	CLA		0100
tion loop and	1002	ADD	A	0051
store	1003	STO	A	0051
Test for equal XRA	1004	TIX (1)	A	1001
Stop	1005	HLT		

Figure 91. Address Modification with Indexing

Another programming aid which permits the changing of an instruction's address is indirect addressing. Bits in positions 12 and 13 denote indirect addressing. They are signified in the instruction format by an "F" and in programs and text by an asterisk following the instruction code (CLA*). One additional computer cycle will be taken whenever indirect addressing occurs. During this cycle the word located at the instruction's address is brought out of storage and its address is used to locate the word upon which the instruction operates. This is sometimes called "the second effective address."

As an example of the feature's use, assume that a word has been read into storage by an input-output device. The programmer knows that the command which read in the data is in location 0100. To bring the data back into the accumulator, a portion of the program could be as shown in Figure 92.

The indirect addressing feature may also be combined with indexing, as mentioned above, to obtain a second effective address.

Location	Instruction	Address	Remarks
0077	XXX	XXXX	Previous command
0100	I-O	-----	Input-output command
	•	•	
0200	CLA*	0100	The CLA* would bring in the data serviced by the I-O instruction even though the address portion is not known.

Figure 92. Indirect Addressing Example

Definition of an Assembly Program

An example of an assembly program is one that defines the symbols and their use as follows:

1. The general format of each instruction is: LOCATION, OPERATION, ADDRESS, TAG, DECREMENT.

   Only those instructions that are referred to by other instructions in the program need be given a symbolic location. All other instructions may be written leaving the location field blank. If the tag and decrement fields are not used, they are left blank. In the case of instructions such as CAQ and VLM, the count is placed in the decrement field. Also, for these instructions, if a tag is not required the instruction would be written in the form OPERATION, ADDRESS, O, COUNT.

2. A symbolic address or location can be composed of one to six alphamerical characters, one of which must be non-numerical. For example: TEMP1, GO, HALT, X1 are all allowable symbols. Thus, each symbol can have an important

mnemonic value. Six special characters may not be used in a symbol. They are + – * / , and $. These characters are used for special operations. For example, the plus sign is used for the addition of two or more symbols and/or numbers. Such an operation might be A1 + A2 or HALT + 3.

3. When dealing with a block of data words, only one location in the block need be assigned a symbol. For example, if a block of data words consisted of A1, A2, . . . A75, the location of the first word of the block could be given the symbol AONE. All other words in the block would be related to this point. If A23 were to be referred to it, would be by the symbol AONE + 22.
4. When the actual value of an address, decrement, or count is known, it should be written in absolute form.

When the program has been written it is prepared for assembly by punching each instruction and piece of data into a separate IBM card. These cards are then referred to as symbolic cards.

This symbolic deck is converted to magnetic tape through the card-to-tape equipment and entered, along with the assembly program, into the computer. If desired, the symbolic program may be entered directly into the computer through the on-line card reader.

Assembly

In the assembly process, the symbolic instructions are processed as follows:

1. The symbolic operation codes are replaced with the actual patterns used by the computer. For example, CLA is replaced by the combination of bits 000 101 000 000 which occupy positions S, 1-11 of the 36-bit instruction word in storage.
2. The absolute location for the first instruction of the program is determined by the programmer and given to the assembly program. Each succeeding instruction and data word is given an absolute location stepped up by one. It is therefore important that the symbolic deck be in the proper order. Each symbolic location detected by the assembly program is entered into a table (called the symbolic table) along with its assigned absolute location. The assembly program then replaces the symbolic address with the absolute locations from the table.

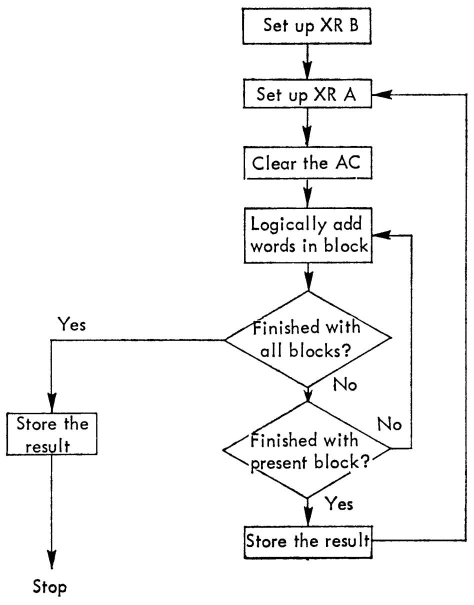
Normally, as a product of the assembly program, a listing of the program in the symbolic format and the actual machine language program is made. In addition, the assembly program furnishes the programmer with a deck of cards containing the machine language program.

Logical Check Sums

One of the principal methods of keeping a check on a block of information in storage is to attach to this block a sum value of all the words in the block. This sum is called the check sum. The best possible check sum that can be formed is one that is developed using the logical operations of the computer. This check sum is known as a logical check sum. It is normally not equal to the algebraic sum of the block. When using a logical check sum there is no possibility of overflow as in the case of algebraic sums. Furthermore, it does not matter in what direction the words of the block are added. This is not true in algebraic summation where overflow possibilities are affected by the direction of summing. An example of the computing of check sums is shown in Figure 93. The programmer knows that there are five blocks with nine words in each block. The first block starts at location 0601, the second at 0611, the third at 0621, and so on. Location 0500 contains a 9 and location 0501 contains a 49. The problem is to find the logical sum of each block and place it in the first location preceding that block.

The coding of a program instruction normally follows this sequence: (1) the location of the instruction, (2) the instruction mnemonic, (3) the address, if any, (4) the index register, if any (5) the decrement. Thus a TIX, 1000, A, 1 would mean that the TIX transfer address is 1000, index register A is to be used, and a decrement value of 1 is involved. The location of the instruction would precede the TIX.

The program shown in Figure 94 will compute the logical check sum for a block of 300 words in core storage. Assume that the 300 words are located in storage in locations 700 through 999. The resulting check sum is to be stored in location 1000. The program uses an index-register-controlled loop to form the logical check sum. The contents of index register 1 are used to effectively modify the address of the instruction in location 102. The index register initially contains the number 300. The final value in the index register will be 1 since the decrement of the TIX instruction is 1. The manner in which the two-instruction loop is performed is as follows:


Set up XR B
Set up XR A
Clear the AC
Logically add words in block
Finished with all blocks?
Yes
No
Finished with present block?
No
Yes
Store the result
Store the result
Stop

LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	
0100	LXA	501,B	49 to XRB
0101	LXA	500,A	9 to XRA
0102	CLM		Clear the accumulator
0103	ACL	650,B	Add the block
0104	TNX	110,B,1	Test all blocks for end
0105	TIX	103,A,1	Reduce count
0106	SLW	640,B	Store the check sum for block
0107	TIX	101,B,1	Test for end of block
0110	SLW	640	Store check sum (last one)
0111	HPR		Stop

Figure 93. Computing Check Sum Program and Flow Chart

LOOP CYCLE	I.R. 1	EFFECTIVE ADDRESS OF ACL INSTRUCTION
End of 1st cycle (Before TIX executed)	300	ACL 700
End of 2nd cycle (Before TIX executed)	299	ACL 701
End of 3rd cycle (Before TIX executed)	298	ACL 702
End of 299th cycle (Before TIX executed)	2	ACL 998
End of 300th cycle (Before TIX executed)	1	ACL 999
End of 300th cycle (After TIX executed)	1	Not executed

Normally a symbolic location is assigned to the block of words. For example, the symbol FIRST could be used to designate the location of the first word of the block. The symbol CKSUM could be used to specify the location where the computed logical check sum is to be stored. The program would then be written as shown in Figure 95.

The number of times the loop is executed is dependent upon the value placed into the index register and the value of the decrement of the TIX instruction. In the preceding example, since XRA contained 300 and the decrement of the TIX instruction is 1, the loop is executed 300 times. If the decrement had been 2, the loop would have been executed 150 times. In this case the logical check sum would have been com-

LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
100	AXT	300,1	LOAD 300 INTO INDEX REGISTER 1	
101	CLM		CLEAR ACCUMULATOR (EXCEPT FOR SIGN)	
102	ACL	1000,1	TWO INSTRUCTION LOOP TO COMPUTE LOGICAL	
103	TIX	102,1,1	CHECK SUM. TIX USED TO TEST END OF LOOP	
104	SLW	1000	STORE LOGICAL CHECK SUM IN LOCATION 1000	

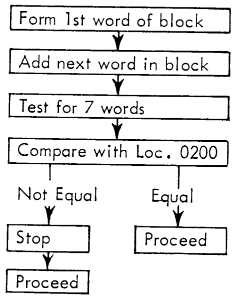
Figure 94. Logical Check Sum Program, Actual

LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
	AXT	300,1	LOAD 300 INTO INDEX REGISTER 1	
	CLM		CLEAR AC (EXCEPT FOR SIGN)	
ADDER	ACL	FIRST + 300,1	TWO INSTRUCTION LOOP TO COMPUTE LOGICAL	
	TIX	ADDER, 1,1	CHECK SUM. TIX USED TO TEST END OF LOOP	
	SLW	CKSUM	STORE LOGICAL CHECK SUM IN LOCATION CKSUM	

Figure 95. Logical Check Sum Program, Symbolic

puted for every other word in the block. Note that at the end of the 300th cycle the index register contained 1. The contents of an index register are never reduced to zero as the result of using a TIX or TNX instruction. The final value found in an index register is dependent on the decrement of the TIX or TNX instruction. If the decrement is the integer K, then, depending upon the initial value of the contents of the index register, the final value of the index register can vary in the range K, K-1, K-2, ... , 3, 2, 1.

One of the ways check sums could be used is shown in Figure 96. The problem is to find the logical sum of a block of seven numbers starting in location 0100. If the sum does not equal the predetermined amount in location 0200, transfer to an error stop. If it does equal the amount in 0200, proceed with the program. The first check sum (original) is in location 0200, and a 6 is in the address part of location 0006.



```
0000    CAL        0100
0001    LXA,   A   0006
0002    ACL,   A   0107
0003    TIX,1, A   0002
0004    ERA        0200
0005    TZE        0007
0006    HPR        0006
0007    ADD        0300
```

Figure 96. Use of Check Sums and Tests

Packing and Unpacking

There are many cases where the information to be handled by the computer is made up of individual items, each of which is less than the size of a computer word. For example, it may be necessary to work with numbers no larger than three decimal digits. To conserve storage space, three such numbers can be stored in the same word as illustrated in Figure 97, where positions S, 14 and 25 are the sign positions of the numbers N_1, N_2, and N_3, respectively. Handling of information in this manner is called "packing." In addition to conserving storage space, packing also increases the entry and exit speed of information by reducing, for instance, the amount of magnetic tape which must be read or written.

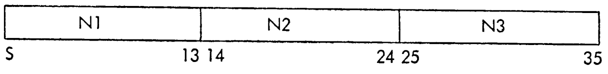


Figure 97. Diagram of Packed Word

Assume that a word in core storage has the form shown in Figure 97, and the number $N2$ is to be operated upon. Before arithmetic operations can be performed with this item, it must be separated from the other data in the word. The method of doing this is called unpacking. The logical operations are employed in this type of operation, as they provide a powerful and flexible tool for carrying out the method. The number $N2$ is to be unpacked from the word without destroying the numbers $N1$ and $N3$. Therefore, the unpacking will be done in the accumulator, saving the packed word in core storage.

The program shown in Figure 98 will accomplish this. The mask used in the program contains 1's in positions 14-24 and 0's elsewhere. The result of using this mask with the ANA instruction will place the number $N2$ in positions 14-24 of the accumulator. By varying the format of the mask, any of the three numbers could have been unpacked (extracted) from the packed word.

H 1	LOCATION 2 6	7	OPERATION 8	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS 72	IDENTIFICATION 73 80
			CAL	PAKWD	PLACE PACKED WORD INTO AC POSITIONS P, 1-35	
			ANA	MASK	N2 LEFT IN AC AS RESULT OF ANA OPERATION	
			ALS	14	SHIFT N2 UNTIL SIGN OCCUPIES POSITION P	
			SLW	LOCN2	STORE N2 IN LOCATION LOCN2	
			.			
	MASK		OCT	000017774000	MASK CONFIGURATION TO OBTAIN N2 ONLY	

Figure 98. Unpacking Program

H	LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
		CAL	MASK	PLACE MASK IN AC POSITIONS P, 1-35	
		ANS	PAKWD	ERASE N2 FROM PACKED WORD	
		CAL	LOCN4	PLACE N4 INTO POSITIONS P, 1-10 OF AC	
		ARS	14	SHIFT N4 INTO POSITIONS 14-24 OF AC	
		ORS	PAKWD	INSERT N4 INTO POSITIONS 14-24 OF LOCATION	
				PAKWD. POSITIONS S, 1-13 AND 25-35 UNCHANGED	
	MASK	OCT	777760003777	MASK TO REMOVE N2 FROM LOCATION PAKWD	

Figure 99. Packing Program

Location	Instruction	Address	Remarks
0000	CLA	0100	Put the number in accumulator
0001	ANA	0050	Extract positions 12-35
0002	STO	0200	Store the result, properly aligned.
0003	HTR		

Figure 100. Masking Program

After performing the desired arithmetic operations on the number *N*2, a new number, *N*4, is the result. This number is the same size as *N*2. Now this new number is to be packed (inserted) in location PADWD replacing *N*2. *N*1 and *N*3 are to remain unchanged. The program in Figure 99 will accomplish this. The program assumes that the number *N*4 occupies positions S, 1-10 of location LOCN4. The mask used with the ANS preserves the numbers *N*1 and *N*3 while replacing N2 with 0's.

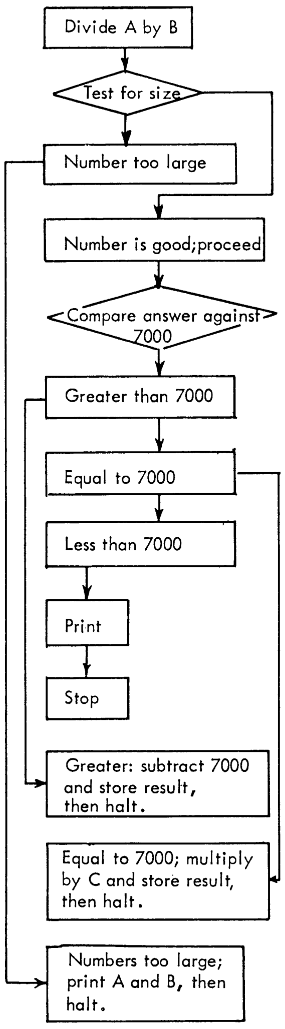
Masking may be used to extract a full number or portion of a word from a given location instead of shifting and adjusting the result. Another example is shown in Figure 100 where a number located in positions 12-35 of location 0100 is to be extracted and stored in location 0200. The mask used is located in 0050 and consists of zeros in positions S-11 and ones in positions 12-35.

The program example in Figure 101 shows a number of test instructions, the compare instruction, and some arithmetic operations. The instruction PRINT means that a print routine is being used and the data being printed are denoted by its address.

The problem is to divide *A* by *B*. If the computer cannot handle the problem, print both *A* and *B*. If the answer equals 7000, multiply it by *C* and save the answer in location 0400. If it is less than 7000, print the answer. If it is more than 7000 put the difference in location 0500.

The programmer is given *A* in location 0100, *B* in location 0101, *C* in location 0102, and 7000 in location 0103.

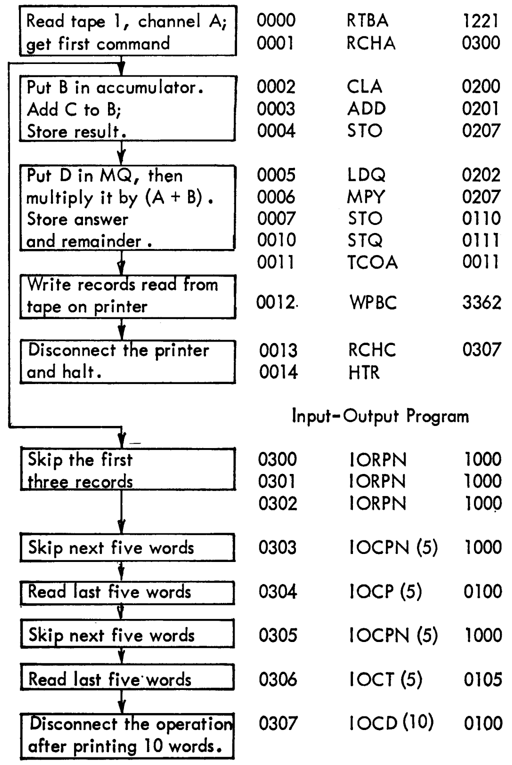
Again the flow chart should serve as an aid in the program steps and is a reference when reading the program.


Divide A by B
Test for size
Number too large
Number is good;proceed
Compare answer against 7000
Greater than 7000
Equal to 7000
Less than 7000
Print
Stop
Greater: subtract 7000 and store result, then halt.
Equal to 7000; multiply by C and store result, then halt.
Numbers too large; print A and B, then halt.

```
0000   LDQ    0100
0001   DVP    0101
0002   DCT

0003   TRA    0022

0004   STQ    0105

0005   CLA    0105
0006   CAS    0103

0007   TRA    0014

0010   TRA    0017

0011   STO    0600

0012   Print  0600

0013   HTR

0014   SUB    0103
0015   STO    0500
0016   HTR

0017   MPY    0102
0020   STO    0400
0021   HTR

0022   Print  0100
0023   Print  0101
0024   HTR
```

Figure 101. Program Example

An example of input-output and computing is shown in Figure 102. There are eight binary records on tape unit 1 attached to channel A. Each record contains ten words. The program should: (1) skip the first three records, (2) skip the first five words of the fourth record, (3) read the last five words of that record, (4) skip the first five words of the fifth record, and (5) read the last five words of the fifth record. Put the ten words read into binary print using the printer attached to channel C. Simultaneously with the reading, solve $(B + C) \times D$ and store the result in 0110 and 0111. B is in location 0200, C is in location 0201, and D in location 0202.


Read tape 1, channel A; get first command
Put B in accumulator. Add C to B; Store result.
Put D in MQ, then multiply it by (A + B). Store answer and remainder.
Write records read from tape on printer
Disconnect the printer and halt.
Skip the first three records
Skip next five words
Read last five words
Skip next five words
Read last five words
Disconnect the operation after printing 10 words.

```
0000   RTBA        1221
0001   RCHA        0300
0002   CLA         0200
0003   ADD         0201
0004   STO         0207
0005   LDQ         0202
0006   MPY         0207
0007   STO         0110
0010   STQ         0111
0011   TCOA        0011
0012   WPBC        3362
0013   RCHC        0307
0014   HTR

      Input-Output Program

0300   IORPN       1000
0301   IORPN       1000
0302   IORPN       1000
0303   IOCPN (5)   1000
0304   IOCP (5)    0100
0305   IOCPN (5)   1000
0306   IOCT (5)    0105
0307   IOCD (10)   0100
```

Figure 102. Simultaneous Read, Write and Compute, then Print

Subroutines

It is very often necessary to repeat the same group of instructions many times during the execution of a program. Examples are the series of instructions necessary for decimal-to-binary conversion, square root, or computing a logical check sum. It is not desirable to write out the necessary instructions each time a function is needed. Instead, the instructions needed are written only once and the main program is then arranged to transfer to this block of instructions each time they are required. Such a block of instructions is called a "subroutine."

These subroutines normally perform such basic functions that they may be used in the solution of many types of problems. For instance, a subroutine which computes a square root can be used in a wide variety of problems. Another example of such a subroutine would be one which computes the logical check sum for a block of words in storage.

Subroutines may be used in two ways with respect to the main program. One method is to insert the subroutine into the main program at the point where it is to be used. Subroutines designed for this type of usage are called "open-subroutines." The open subroutine is "sandwiched" into a program as though it were part of the original coding of the program. This type of subroutine usage is normally restricted to the cases where the main program uses the subroutine only once.

When the main program uses a subroutine several times, which is the common situation, it is apparent that the open subroutine is not desirable. Here, the second method of employing subroutines is used. The subroutine used in these situations is called a "closed subroutine." A closed subroutine may occur several times within one main program, but the set of instructions comprising the subroutine need appear only once. The transfer of control from the main program to the subroutine takes place from a set of instructions known as the calling sequence or basic linkage. The calling sequence transfers control to the subroutine, tells the subroutine where to return to the main program, and gives the subroutine any other information required (Figures 103 and 104).

The subroutine illustrated computes the logical check sum for a block of words in core storage. Three parameters are needed by this subroutine. There are the initial location of the block, the number of words in the block, and the location for storing the resulting check sum. The subroutine then returns control to the main program at the instruction following the last parameter of the calling sequence.

The calling sequence is of the form shown in Figure 103. The subroutine is of the form shown in Figure 104.

The complete transfer of control between the main program and the subroutine is based upon the TSX instruction. As the result of the execution of this instruction, the twos complement of the location LINK is placed in index register 4. From the standpoint of algebraic operation the twos complement of a number is equivalent to the negative of the number. For example, 1 minus (twos complement of LINK) is equivalent to 1 minus (minus LINK) = 1 + LINK.

H	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
1	2 6	7	8		72	73 80
	LINK		TSX	BLKSM,4	AUTOMATIC LINKING INSTRUCTION	
				FIRST	LOCATION OF FIRST WORD IN BLOCK	
				N	NUMBER OF WORDS IN BLOCK	
				CKSUM	LOCATION FOR STORING CHECK SUM	
	---		---	-------	LOCATION TO WHICH CONTROL WILL BE RETURNED	

Figure 103. Calling Sequence

H	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
1	2 6	7	8		72	73 80
	BLKSM		CLA	2,4	GET NUMBER OF WORDS IN BLOCK	
			PAX	0,1	PLACE N IN INDEX REGISTER 1	
			ADD	1,4	ADD LOCATION FIRST TO FORM FIRST +N	
			STA	ADDER	INITIALIZE LOGICAL ADD INSTRUCTION	
			CLA	3,4	GET LOCATION TO STORE CHECK SUM	
			STA	STSUM	PLACE ADDRESS IN STORE INSTRUCTION	
			CLM		CLEAR AC	
	ADDER		ACL	0,1	TWO INSTRUCTION LOOP FOR COMPUTING LOGICAL	
			TIX	ADDER,1,1	CHECK SUM FOR BLOCK OF N WORDS	
	STSUM		SLW	0	STORE CHECK SUM IN LOCATION CKSUM	
			TRA	4,4	RETURN CONTROL TO MAIN PROGRAM	

Figure 104. Subroutine to Compute Logical Check Sum

In the subroutine the instructions which make use of this property are:

INSTRUCTION	EFFECTIVE EXECUTION	EQUIVALENT EXECUTION
CLA 2,4	CLA 2 — (2's comp. LINK)	CLA LINK + 2
ADD 1,4	ADD 1 — (2's comp. LINK)	ADD LINK + 1
CLA 3,4	CLA 3 — (2's comp. LINK)	CLA LINK + 3
TRA 4,4	TRA 4 — (2's comp. LINK)	TRA LINK + 4

From the above table it can be seen that the subroutine will be able to make use of the information found in the parameter locations of the calling sequence without knowledge of their exact location in storage. Since LINK is a symbol representing any location in core storage, the subroutine can thus communicate with the main program at any location in the main program. By means of the TRA 4,4 instruction, the subroutine has the ability to transfer control back to the proper location in the main program.

One of the main responsibilities of a subroutine is to insure that when control is transferred back to the main program the status of all the registers is the same as when control was transferred to the subroutine. This does not apply, of course, to a subroutine designed specifically to change a machine condition. For example, in the previous illustration the contents of index register 1 are destroyed by the subroutine. Thus, when control is transferred back to the main program, index register 1 may not be the same as when control was transferred to the subroutine. The contents of index register 1 may be preserved by adding the instruction SXA SAVE, 1 just after the instruction CLA 2, 4. The contents of XR 1 will be stored in the address part of location SAVE. Now, if location SAVE is inserted just before the TRA 4,4 instruction and contains the instruction AXT 0,1, the original contents of XR 1 will be replaced just before control is transferred back to the main program.

Convert Instructions

Three convert instructions are available in the computer. During their execution these instructions use, in addition to core storage, the accumulator, multiplier-quotient, and storage registers. Index register 1 may also be used, if desired, to receive information at the conclusion of a convert instruction execution. These instructions normally work with tables stored in core storage.

These convert instructions provide the programmer with a rapid means of performing such operations as

BCD-to-binary and binary-to-BCD conversion, BCD arithmetic, editing of records, and modification of collating sequences.

When the convert instructions are used, either the AC or the MQ contains a 36-bit word that is divided into six 6-bit binary numbers. Each 6-bit number (sometimes referred to as a character) is treated separately in consecutive order by the convert instructions. For the instructions CRQ and CVR, this 36-bit word represents the actual word operated upon. The CVR examines the word six bits at a time from right to left, while CRQ examines the word six bits at a time from left to right. For the CAQ, the contents of the MQ are examined six bits at a time from left to right while addition of quantities found in core storage, as determined by this word in the MQ, takes place in the AC.

The problem of replacing the leading zeros of a BCD number with blanks is reduced to a short rapid program through the use of the CRQ instruction. This particular convert instruction is used because, to remove leading zeros, the BCD number must be tested from left to right.

To carry out the editing (modification) of the BCD number, it is necessary to set up a table in core storage. This table has the following format:

LOCATION	CONTENTS S,1 − 5	CONTENTS 21 − 35
A	BCD blank (b)	A
A + 1	BCD one	A + 10
A + 2	BCD two	A + 10
.	.	.
A + K	BCD (K)	A + 10
.	.	.
A + 9	BCD nine	A + 10
A + 10	BCD zero	A + 10
A + 11	BCD one	A + 10
A + 12	BCD two	A + 10
.	.	.
A + 19	BCD nine	A + 10

The program shown in Figure 105 will perform the required editing operation on a BCD number of 12 digits occupying two consecutive core storage locations. The program must consider the following cases:

1. All leading zeros must be sensed and replaced by blank characters.
2. All non-zero digits must be preserved.
3. Once a non-zero digit is found, all succeeding zeros must be preserved.
4. If a non-zero digit is found in the high-order six digits, the second half of the number need not be processed.

The program is executed as follows: The high-order six digits of the BCD number are placed in the MQ by the LDQ BCD1 instruction. The first table reference made by the CRQ A,1,6 instruction will be at location A + *N*, where *N* is the high-order digit in the MQ, that is, $c(\text{MQ})_{S,1\text{-}5}$. If *N* is zero, the CRQ instruction will go to table location A and from this location will replace the zero with the BCD blank character. Since the address part of location A contains A, the second table reference will begin at location A of the table. Once a non-zero digit occurs, the CRQ instruction will make a table reference at location A + *K*, where *K* is the non-zero digit. Location A + *K* contains *K* in positions S,1-5 and A + 10 in the address part. Thus, the value *K* will replace the number *K* and this insures that the non-zero digits will be preserved. Once the first non-zero digit is found, only the second part of the table, location A + 10 through A + 19, is used. This insures that zeros following the first non-zero will be replaced with zeros instead of blanks. Figure 106 illustrates the execution of the convert instruction using the BCD number 000307.

When all six of the BCD digits have been tested (the count reduced to zero), the execution of the CRQ instruction is terminated. Since the instruction contains a tag of one, the address part of the last table reference location will be placed in index register 1. After

H 1	LOCATION 2–6	7	OPERATION 8	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS 72	IDENTIFICATION 73–80
	START		LDQ	BCD1	LOAD HIGH ORDER SIX DIGITS INTO MQ	
			CRQ	A,1,6	EDIT HIGH ORDER SIX DIGITS	
			STQ	BCD1	STORE EDITED DIGITS	
			TXH	OUT,1,A	TEST FOR NON-ZERO THIS HALF	
			LDQ	BCD2	LOAD LOW ORDER SIX DIGITS INTO MQ	
			CRQ	A,0,6	EDIT LOW ORDER SIX DIGITS	
			STQ	BCD2	STORE EDITED LOW ORDER DIGITS	
	OUT		---	--------	PROGRAM CONTINUES HERE	

Figure 105. Edit Program

storing the edited BCD number (STQ BCD1) the contents of index register 1 are then compared with the number A by the TXH OUT,1,A instruction. If the index register contains A, then all six of the high-order BCD digits were zero and the program will continue to examine the remaining digits in the number. If the index register contains A + 10, this indicates that a non-zero digit was found in the high-order six digits. Thus, the low-order digits need not be processed and control is transferred to location OUT.

The convert instruction CVR can be used to perform BCD addition without having to rely on a complex logical routine. The CVR instruction is used in this application since it is necessary to process the decimal sum from right to left to provide for carries from one position to the next. The program which will add two 6-digit unsigned BCD numbers is shown in Figure 107. This program insures that the following conditions are satisfied.

1. Any position of the sum that does not produce a carry must be preserved.
2. Any position of the sum that does produce a carry must be modified with the carry propagated to the next position.
3. A test must be made to determine whether or not a carry occurs out of the high-order position. If such a carry does occur, a one must be placed in the next highest word location.

C(MQ)						Count	C(SR)		C(MQ) + C(SR) = X			C(X)		Remarks
S,1-5	6-11	12-16	17-23	24-29	30-35		S,1-5	21-35	S,1-5	21-35		S,1-5	21-35	
0	0	0	3	0	7	6	—	A	0	A	A+0	b	A	Start cycle 1
0	0	3	0	7	b	5	b	A						End cycle 1
0	0	3	0	7	b	5	b	A	0	A	A+0	b	A	Start cycle 2
0	3	0	7	b	b	4	b	A						End cycle 2
0	3	0	7	b	b	4	b	A	0	A	A+0	b	A	Start cycle 3
3	0	7	b	b	b	3	b	A						End cycle 3
3	0	7	b	b	b	3	b	A	3	A	A+3	3	A+10	Start cycle 4
0	7	b	b	b	3	2	3	A+10						End cycle 4
0	7	b	b	b	3	2	3	A+10	0	A+10	A+10	0	A+10	Start cycle 5
7	b	b	b	3	0	1	0	A+10						End cycle 5
7	b	b	b	3	0	1	0	A+10	7	A+10	A+17	7	A+10	Start cycle 6
b	b	b	3	0	7	0	7	A+10						End cycle 6

Figure 106. Execution of CRQ

H 1	LOCATION 2 6	7	OPERATION 8	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS 72	IDENTIFICATION 73 80
			CAL	DEC1	FIRST UNSIGNED BCD NUMBER TO AC	
			ADD	DEC2	ADD SECOND BCD NUMBER, SUM IN AC	
			CVR	A, 1, 6	REPLACE VALUES FOR WHICH CARRIES OCCURED	
			SLW	SUM	STORE SUM	
			TXL	OUT, 1, A	TEST FOR HIGH ORDER CARRY	
			CLA	LOCONE	CARRY OCCUPIED, PLACE BCD ONE IN AC	
			STO	SUM + 1	PLACE HIGH ORDER CARRY IN NEXT LOCATION	
	OUT		----	--------	PROGRAM CONTINUES HERE	
			:			
	LOCONE		HTR	1		

Figure 107. BCD Addition Program

LOCATION	BCD CHARACTER POSITIONS S-5	NEXT TABLE REFERENCE POSITIONS 21-35
A	0	A
A + 1	1	A
A + 2	2	A
.	.	.
.	.	.
A + 9	9	A
A + 10	0	A + 1
A + 11	1	A + 1
A + 12	2	A + 1
.	.	.
.	.	.
A + 19	9	A + 1

Figure 108. Table for BCD Addition

The convert instruction CVR used in the program uses the table found in Figure 108.

The execution of this program can best be illustrated by the following example. The two BCD unsigned numbers 434589 and 691593 are to be added. The resulting sum is considered as six 6-bit numbers (Figure 109). The low-order 6-bit number has the value 12. Thus, the first table reference made by the CVR A,1,6 instruction is at location A + 12. Positions S,1-5 of this location contain a BCD 2 which replaces the number 12 in the AC. Positions 21-35 of this location contain the number A + 1. The address A + 1 causes the next table reference to be made at location A + 18 rather than A + 17. Thus, a carry of one is propagated from the units to the tens position of the sum. The execution of the CVR will end when the count has been reduced to zero. Since this instruction has a tag of one, the contents of the storage register

Location	Binary equivalent of BCD Digit. Positions S, 1-19	Next table location Positions 21-35
A	0	B
A + 1	1×10^5	B
A + 2	2×10^5	B
.	.	.
.	.	.
A + 9	9×10^5	B
B	0	C
B + 1	1×10^4	C
B + 2	2×10^4	C
.	.	.
.	.	.
B + 9	9×10^4	C
C	0	D
C + 1	1×10^3	D
C + 2	2×10^3	D
.	.	.
.	.	.
C + 9	9×10^3	D
D	0	E
D + 1	1×10^2	E
D + 2	2×10^2	E
.	.	.
.	.	.
D + 9	9×10^2	E
E	0	F
E + 1	1×10	F
E + 2	2×10	F
.	.	.
.	.	.
E + 9	9×10	F
F	0	0
F + 1	1	0
F + 2	2	0
.	.	.
.	.	.
F + 9	9	0

Figure 110. Table for BCD-to-Binary Conversion

		ACCUMULATOR CONTENTS						STORAGE REGISTER			
								START OF CYCLE		END OF CYCLE	
INSTRUCTION	COUNT	P - 5	6-11	12-17	18-23	24-29	30-35	S - 5	21 - 35	S - 5	21 - 35
CAL DEC1		4	3	4	5	8	9				
ADD DEC2		10	12	5	10	17	12				
CVR A,1,6	6	10	12	5	10	17	12	–	A+12	2	A+1
	5	2	10	12	5	10	17	2	A+1+17	8	A+1
	4	8	2	10	12	5	10	8	A+1+10	1	A+1
	3	1	8	2	10	12	5	1	A+1+5	6	A
	2	6	1	8	2	10	12	6	A+12	2	A+1
	1	2	6	1	8	2	10	2	A+1+10	1	A+1
	0	1	2	6	1	8	2	1	A+1		

Figure 109. Execution of CVR

positions 21-35 (A + 1) will be placed in index register 1. The TXL OUT,1,A instruction then tests the contents of this index register. Since A + 1 is greater than A, the program continues and a BCD 1 is placed in positions 30-35 of location SUM + 1. Had index register 1 contained A, the program would have transferred control to location OUT.

Conversion from one number system to another may be performed by the CAQ convert instruction. An example of BCD-to-binary conversion is illustrated here. The program which performs this conversion is based upon the fact that a BCD number (e.g., 803157) is really a sum of terms of the form:

$$8 \times 10^5 + 0 \times 10^4 + 3 \times 10^3 + 1 \times 10^2 + 5 \times 10 + 7.$$

The binary number equivalent to a BCD number is obtained simply by finding the sum of the binary equivalents of each term. For this example, the binary equivalent of 8×10^5 plus the binary equivalent of 0×10^4 plus . . . plus the binary equivalent of 7. The table used with this program is thus divided into six parts (Figure 110). Each section consists of ten words, one word for each of the digits 0-9 multiplied by a power of ten. The binary equivalent of each BCD digit is contained in the twenty positions S-19 of each word in the table. This is based on the fact that only 20 binary positions are necessary to represent a 6-digit BCD number.

The CAQ instruction uses a straightforward table look-up operation to build up the binary equivalent of the BCD number, digit by digit. Each digit of the BCD number is employed to look up its binary equivalent from one of the six parts of the table. The first BCD digit will choose its binary equivalent from the first section (10^5) of the table; the second BCD digit will choose its equivalent from the second section of the table; and so forth. This operation is continued until the complete binary equivalent is formed in positions P-19 of the AC. Caution must be taken to choose table locations so that the sum of the locations (as illustrated in Figure 112) will not overflow into the resulting binary number portion of the AC.

The program in Figure 111 will perform the conversion operation.

The execution of the CAQ A,0,6 instruction is illustrated in Figure 112. As can be seen, the sum of the table locations B,C, . . .,F must not form a sum that will overflow into the binary portion of the AC.

H 1	LOCATION 2-6	7	OPERATION 8	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS 72	IDENTIFICATION 73-80
			LDQ	BCDWD	LOAD BCD WORD INTO MQ	
			CLM		CLEAR AC (EXCEPT FOR SIGN)	
			CAQ	A, 0, 6	CONVERT BCD TO BINARY	
			ARS	16	SHIFT BINARY RESULT TO PROPER POSITION	
			SLW	BINWD	STORE BINARY RESULT	

Figure 111. BCD-to-Binary Conversion Program

INSTRUCTION	COUNT	ACCUMULATOR CONTENTS (Binary equivalent) P,1 - 19	21 - 35	MQ CONTENTS	STORAGE REGISTER START OF CYCLE S - 19	21-35	END OF CYCLE S - 19	21 - 35
LDQ BCDWD				803157				
CLM		0000.....................000	00......000	803157				
CAQ A,0,6	6				–	A + 8	$8\text{x}10^5$	B
	5	$8\text{x}10^5$	B	031578	$8\text{x}10^5$	B + 0	0	C
	4	$8\text{x}10^5+0$	B+C	315780	0	C + 3	$3\text{x}10^3$	D
	3	$8\text{x}10^5+0+3\text{x}10^3$	B+C+D	157803	$3\text{x}10^3$	D + 1	$1\text{x}10^2$	E
	2	$8\text{x}10^5+0+3\text{x}10^3+1\text{x}10^2$	B+C+D+E	578031	$1\text{x}10^2$	E + 5	5x10	F
	1	$8\text{x}10^5+0+3\text{x}10^3+1\text{x}10^2+5\text{x}10$	B+C+D+E+F	780315	5x10	F + 7	7	0
	0	$8\text{x}10^5+0+3\text{x}10^3+1\text{x}10^2+5\text{x}10+7$	B+C+D+E+F	803157	7	0		

Figure 112. Execution of CAQ

Sense Indicators

In many applications of data-processing machines it is desirable to have switches which can be set and tested by the program. A special register, the sense indicator register, provides 36 such devices. Each of these can be turned on or off (zero or one). The individual positions are called sense indicators.

Either singly or in groups these indicators can be turned on or set (set to ones) or turned off or reset (set to zeros). These indicators can be used as program-controlled sense switches, sense lights, or selectors. In addition, they are also useful in extracting or inserting parts of words and for testing fields within a word.

A program is often written that deals with a problem having several variations. In cases of this type, one way of developing the general program is to construct the program so that it is composed of self-contained sections. Control in the general program is then transferred from section to section in the sequence determined by the particular variation being solved. The sense indicators may be used to direct and monitor the desired sequence of control.

For example, a general program is composed of eight sections and is to deal with a problem having four variations. The eight sections and the four variations are illustrated in Figure 113. The sequence of control necessary for each variation and the representation of the sense indicator bits needed for the direction of this sequence are shown in Figure 114.

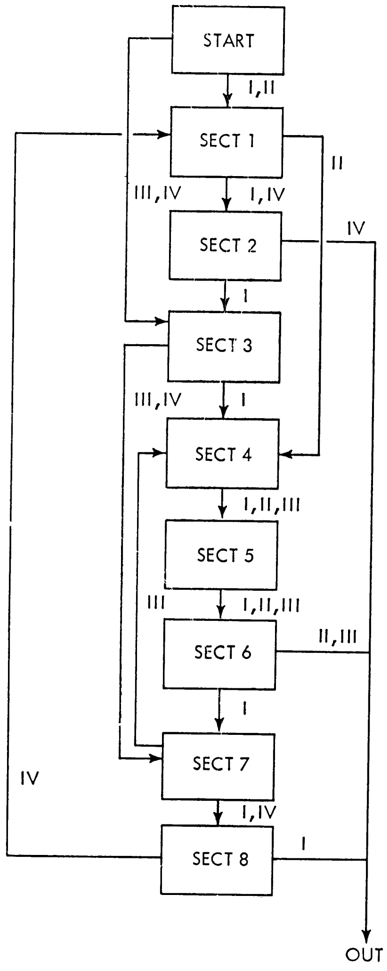


Figure 113. Block Diagram of General Program

VARIATION	SEQUENCE OF CONTROL BY PROGRAM SECTION	SENSE INDICATOR REGISTER POSITIONS 32	33	34	35
I	1-2-3-4-5-6-7-8	0	0	0	1
II	1-4-5-6	0	0	1	0
III	3-7-4-5-6	0	1	0	0
IV	3-7-8-1-2	1	0	0	0

Figure 114. Sense Indicator Pattern

Four control words are used to contain the different bit patterns to be used in the sense indicator register for the four different problem variations (Figure 115). The general program is started by loading the desired control word into the sense indicator register.

LOCATIONS	CONTROL WORD	REMARKS
VARI	Oct 1	Variation 1 bit pattern for the SI register
VARII	Oct 2	Variation 2 bit pattern for the SI register
VARIII	Oct 4	Variation 3 bit pattern for the SI register
VARIV	Oct 10	Variation 4 bit pattern for the SI register

Figure 115. Problem Variations

The instructions in the general program that are necessary for the direction of control are shown in Figure 116.

Both the RFT and RNT instructions contain masks (in octal) in positions 18-35. These 18 bits are compared with the right-most (positions 18-35) 18 bits of the sense indicator register. The instruction LFT and LNT with the same masks could have been used in place of the RFT and RNT instructions. However, since these instructions compare with the left-most

H	LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
	START	LDI	CONWD	LOAD CONTROL WORD FOR DESIRED SEQUENCE	
		RFT	14	INITIAL SEQUENCE TEST. SI POSITIONS 32 AND 33	
		TRA	SECT3	TESTED FOR ZEROS. IF ZEROS CONTROL TO SECT1. IF	
	SECT1			NOT CONTROL TO SECT 3	
		RFT	2	SECTION 1 END-SEQUENCE TEST. TEST POSITION 34.	
		TRA	SECT4	IF ZERO CONTROL GOES TO SECTION 2. IF ONE,	
	SECT2			CONTROL TO SECTION 4.	
		RFT	10	SECTION 2 END-SEQUENCE TEST. TEST POSITION 32.	
		TRA	OUT	IF ZERO CONTROL GOES TO SECTION 3. IF ONE, END-	
	SECT3			OF-PROGRAM.	
		RFT	14	SECTION 3 END-SEQUENCE TEST. TEST POSITIONS 32	
		TRA	SECT7	AND 33. IF ZERO, CONTROL GOES TO SECTION 4. IF	
	SECT4			ONES, CONTROL TO SECTION 7.	
	SECT5			CONTROL ALWAYS GOES FROM SECTION 4 TO SECTION	
				5.	
	SECT6			CONTROL ALWAYS GOES FROM SECTION 5 TO SECTION	
				6.	
		RNT	1	SECTION 6 END-SEQUENCE TEST. TEST POSITION 35.	
		TRA	OUT	IF ONE, CONTROL GOES TO SECTION 7. IF ZERO, END-	
	SECT7			OF-PROGRAM.	
		RFT	4	SECTION 7 END-SEQUENCE TEST. TEST POSITION 33.	
		TRA	SECT4	IF ZERO, CONTROL GOES TO SECTION 8. IF ONE,	
	SECT8			CONTROL TO SECTION 4.	
		RFT	10	SECTION 8 END-SEQUENCE TEST. TEST POSITION 32.	
		TRA	SECT1	IF ONE, TRANSFER CONTROL TO SECTION 1. IF ZERO,	
	OUT			END-OF PROGRAM.	

Figure 116. Control Instructions in Program

18 positions (positions 0-17) of the sense indicator register, the four commands must be changed so that the four SI positions concerned are positions 14-17.

It may be necessary to unpack a word without affecting the accumulator or the multiplier-quotient register. In this case, the sense indicator register may be used for the unpacking operation. For example, to unpack a number occupying positions 14-24 of a packed word, a mask containing ones in positions S,1-13,25-35 is used to perform the extraction.

Figure 117 is the program which is used to execute the extracting. The packed word is loaded into the SI register by the LDI instruction. With the execution of the RIS instruction all the positions in the SI corresponding to the ones in the mask are set to zero. Thus, as a result of the mask used, the desired number in positions 14-24 is left intact and the remaining positions are set to zero. The number extracted from the packed word is simply dependent on the mask used.

A companion operation to the extracting described above is that of insertion. This may also be accomplished by using the SI register. An example is inserting a new number in positions 14-24 of the packed word of the last example. The new number to be inserted occupies positions 14-24 of the AC. The program is shown in Figure 118. The packed word is loaded into the SI by the LDI instruction. The mask used by the RIS instruction sets positions 14-24 of the SI to zero. The remaining positions are unchanged. The new number is then "OR'ed" into the packed word by the OAI instruction.

Floating Point Overflow and Underflow

During many scientific and engineering problems, the programmer is faced with the difficulty of keeping track of the decimal point. To aid in this respect, the computer is equipped with a complete set of floating point instructions. Briefly, a floating point number is treated as a 27-bit signed proper fraction and an 8-bit characteristic which represents a signed exponent.

By the nature of floating point operation, the fraction may never overflow the registers, and an underflow of the fraction produces a normal zero which is a proper result. The characteristic enjoys no such freedom. A floating point operation resulting in a characteristic, either too large or too small for that portion of the word set aside for it, produces a condition known as *floating point overflow or underflow*, respectively. These conditions are referred to collectively as *floating point spill.*

When floating point spill occurs, the factors must be scaled to fall within the range of the computer registers if calculation is to continue. The exact details of this scaling usually depend upon the conditions of the problem. However, the computer provides adequate facilities to assist the programmer in deciding what these conditions are and for controlling the corrective process.

The computer may be operated in two modes with regard to floating point operation. Normally, the computer operates in *floating point trap mode;* that is, the floating point trap device is normally on. This device is referred to as "FPT." If the instruction leave floating trap mode (LFTM) is executed, the computer operates in what is called the 704 floating point trap mode.

IBM 704 Floating Point Trap Mode

To enter the 704 floating point trap mode, the instruction LFTM must be executed. It is necessary be-

H	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
1	2 6	7	8		72	73 80
			LDI	PAKWD	LOAD PACKED WORD INTO SI	
			RIS	MASK	MASK TO PRESERVE POSITIONS 14-24 OF SI	
			STI	UNPAK	STORE UNPACKED NUMBER INTO STORAGE	
			⋮			
	MASK		OCT	777760003777	MASK TO OBTAIN POSITIONS 14-24	

Figure 117. Extraction Program Using Sense Indicator

H	LOCATION		OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
1	2 6	7	8		72	73 80
			LDI	PAKWD	LOAD PACKED WORD INTO SI	
			RIS	MASK	MASK TO CLEAR POSITIONS 14-24	
			OAI		OR NEW NUMBER INTO POSITIONS 14-24	
			STI	PAKWD	STORE NEW PACKED WORD	
			⋮			
	MASK		OCT	000017774000	MASK TO CLEAR POSITIONS 14-24	

Figure 118. Insertion Program Using Sense Indicator

cause FPT is normal in the computer. If a floating point spill occurs while in the 704 mode, the accumulator and/or the MQ overflow indicators are turned on. Which indicator is set is determined by the register producing the spill. Each indicator may be tested by the program, subsequent to the spill, by executing the proper overflow test instruction. The mathematics involved in the following examples are specialized to point out how the logical facilities of the computer might be employed to detect and to control the correction of floating point spill.

Floating Point Spill in the 704 Mode

This problem is defined below.

A body of surface area (SURA) is being bombarded by particles of some nature. The researcher has measured the number of impacts (IMP) on the surface at N discreet time intervals. At a certain point in the calculations, the programmer wishes to know if the following conditions exist: $Q < MAX$, where MAX is a constant and Q is the average number of impacts per unit area. If the inequality holds, the program is to continue at location OK. Otherwise, program control is to be transferred to symbolic location TOBIG.

All the IMP_n (total impacts in the N time intervals) are written within the range $0 \leqq IMP_n < 10^{45}$. The IMP_n are stored at symbolic locations IMP to IMP + N − 1. The values N, MAX, and SURA are known. The original values of IMP_n must remain intact. The first step is to calculate the total impacts. If a spill occurs, the IMP_n are scaled by 2^{-100}, and the summation repeats. If another spill occurs, it is treated as an irretrievable error. Note that MQ spill has no effect on this summation. The program is shown in Figure 119 and executed as described below:

Line 1. LFTM must be given to enter the 704 mode. The accumulator overflow indicator is then turned off by the TOV instruction; this is necessary because its condition cannot be assumed.

Line 6. The IMP_n is brought to the accumulator. Because of the method of scaling used, zero words are skipped over.

Line 8. Because spill detection is most useful if the first spill is detected, scaling is accomplished without using floating point. If XRA is zero, scaling is not required and the program proceeds to line 11. If XRA is not zero, scaling is required. A word containing an octal 100 in the characteristic field is fixed-point subtracted from the IMP_n. This is equivalent to floating point division by 2^{100}. If the accumulator goes minus, the IMP_n is not within the given range. If IMP_n is not in this range, control is transferred to 1ERR.

Line 11. The partial sum is added to the IMP_n in the accumulator. If spill does not occur, the program proceeds with the summation; if spill does occur, XRA is set to 1 and the summation is restarted. If scaling has already occurred, an error is indicated and control transfers to 2ERR.

Line 18. The MQ overflow and the divide check indicators are reset to the off position, because their condition cannot be assumed. The FDP is then executed and both the divide check and MQ overflow indicators are tested (MQ overflow or underflow is defined as a TOBIG condition).

Line 27. With the quotient in the accumulator and the condition of the accumulator overflow indicator established (lines 25 and 26), a test to find if scaling has taken place is executed. If so, the word subtracted in line 9 is added back to increase the characteristic by 100, which is equivalent to floating point multiplication by 2^{100}. An overflow at this point (fixed point overflow) indicates that Q is too big. If scaling has not occurred, control transfers.

Line 30. MAX is then subtracted from the quotient. A spill at this point is not significant; the sign of the result establishes the range of the number. If the accumulator is minus, the program proceeds to symbolic location OK, where FPT is reset to the on position and the main program may resume.

Floating Point Trap

Figure 119 illustrates how the interpretation of spill depends upon the conditions prevailing when the spill is detected. In general, the program should: (1) detect the spill as soon as it occurs, (2) know in which register the spill occurred, and (3) know what instruction was being executed when the spill occurred. This information can be provided automatically by the floating point trap.

When spill occurs with the FPT on, the computer automatically performs the following steps:

1. The address plus 1 of the instruction causing spill is placed in the address field of core location 0000.
2. A four-bit code which identifies the nature of the spill is placed in positions 14 through 17 of core location 0000.
3. The computer takes its next instruction from location 0010 and proceeds from there.

To illustrate a use of FPT, the problem used in Figure 119 is repeated using the floating-point-trap feature. Note that, although the program is larger, the possibilities for control are increased. Figure 120 shows that the trap routine starting at location 0010 can conditionally return to the main program or initiate a general operation (in this case, a print program and then halt).

The practice of continually storing certain addresses from the main program in a standard routine is sometimes referred to as "breakpoint" programming; as Figure 120 shows, it is an extremely powerful and

PROBLEM				
CODER		DATE	PAGE	OF

LOCATION	OPERATION	ADDRESS, TAG, DECREMENT/COUNT	COMMENTS	IDENTIFICATION
TIMP	LFTM		TURN OFF FPT.	
	TOV	*+1	TURN OFF ACC OVERFLOW.	
	AXT	0,1	INITIALIZE.	
	AXT	N,2		
	STZ	FREE		
	CLA	IMP+N,2	GET FIRST MEASUREMENT.	
	TZE	TIMP+10	IF ZERO, GET NEXT MEASUREMENT.	
	TXL	*+3,1,0	IS SCALING NECESSARY?	
	SUB	2HND	YES, DIVIDE BY 2^{100}.	
	TMI	1ERR	ALL IMP MUST BE $>$ 2^{-200}.	
	FAD	FREE	ADD.	
	TNO	*+3	DID SPILL OCCUR?	
	TXH	2ERR,1,0	YES, IS THIS THE FIRST SPILL?	
	TXI	TIMP+3,1,1	YES, SET INDICATION AND RESTART.	
	TNX	DONE,2,1	IS SUM COMPLETE? IF	
	STO	FREE	NOT, STORE PARTIAL SUM AND	
	TRA	TIMP+5	GET NEXT MEASUREMENT.	
DONE	TQO	*+1	TURN OFF MQ OV INDICATOR.	
	DCT		TURN OFF DIVIDE CHECK INDICATOR.	
	NOP			
	FDP	SURA	DIVIDE BY SURFACE AREA.	
	DCT		TEST.	
	TRA	TOBIG	COULD NOT DIVIDE.	
	TQO	TOBIG	QUOTIENT OUT OF RANGE.	
	TOV	*+1	TURN OFF ACC OVERFLOW INDICATOR.	
	XCA		QUOTIENT TO THE ACC.	
	TXL	*+3,1,0	WAS TIMP SCALED?	
	ADD	2HND	YES, --MULTIPLY BY 2^{100}.	
	TOV	TOBIG	FIXED POINT OVERFLOW INDICATES N TOO LARGE.	
	FSB	MAX	SUBTRACT.	
	TPL	TOBIG	$Q \geq$ MAX.	
	TRA	OK	Q IS OK, PROCEED.	
FREE	BSS	1		
2HND	OCT	100000000000		
OK	EFTM		TURN ON FPT.	
	PROCEED WITH MAIN PROGRAM.			

Figure 119. Floating-point Spill, 704 Mode

PROBLEM

CODER | DATE | PAGE | OF

```
H  LOCATION   OPERATION  ADDRESS, TAG, DECREMENT/COUNT    COMMENTS                                  IDENTI-
                                                                                                     FICATION
1 2       6 7 8                                                                                  72 73   80
 1  TIMP      STZ                                         CLEAR.
 2            AXT         SPILL,1                         SET CONTROL ADDRESS IN THE
 3            SXA         8,1                             FPT CONTROL ROUTINE.
 4            AXT         0,2                             INITIALIZE
 5  ADD       AXT         N,1
 6            STZ         FREE
 7            CLA         IMP+N,1                         FIRST MEASUREMENT.
 8            TXL         *+4,2,0                         IS SCALING NEEDED?
 9            TZE         *+3                             YES, BUT SKIP ZEROS.
10            SUB         2HND                            DIVIDE BY 2^100.
11            TMI         1ERR                            SHOULD NOT BE MINUS.
12            FAD         FREE                            ADD
13            STO         FREE                            STORE PARTIAL SUM.
14            TIX         *-7,1,1                         CONTINUE
15            TRA         PROC                            SUM COMPLETE, PROCEED.
16  SPILL     TXI         *+1,2,1                         GET CONTROL, SET SCALE SIGNAL.
17            STO         FREE                            SAVE PARTIAL SUM
18            CAL         0                               GET WORD AT LOCATION 0000.
19            ARS         19                              MQ UNDERFLOW IS NOT SIGNIFICANT.
20            TNZ         *+2                             ACCUMULATOR SPILL IF NOT ZERO.
21            TXI         SPILL-2,2,32767                 MAKE XRB  = 0 AND PROCEED.
22            TXH         2ERR,2,1                        ERROR IF A SECOND SPILL.
23            TRA         ADD                             RESTART SUM WITH SCALING
24  PROC      AXT         SPIL2,1                         NEW CONTROL ADDRESS FOR
25            SXA         8,1                             FPT CONTROL ROUTINE.
26            DCT                                         TURN INDICATOR OFF.
27            NOP
28            FDP         SURA                            DIVIDE BY SURFACE AREA.
29            DCT                                         TEST DIVIDE CHECK INDICATOR.
30            TRA         TOBIG                           NUMBER IS OUT OF RANGE.
31            TOV         *+1                             TURN INDICATOR OFF.
32            XCA                                         MQ TO ACCUMULATOR.
33            TXL         *+3,2,0                         WAS MQ SCALED?
34            ADD         2HND                            YES, MULTIPLY BY 2^100.
35            TOV         TOBIG                           NUMBER IS OUT OF RANGE.
36            FSB         MAX                             CHECK MAX.
37            TPL         TOBIG                           NUMBER IS OUT OF RANGE IF PLUS.
38            TRA         OK                              CONTINUE.
39  FREE      BSS
40  2HND      OCT         100000000000
41  OK        AXT         OK,1                            NEW CONTROL ADDRESS.
42            SXA         8,1
43            PROCEED WITH MAIN PROGRAM
44  SPIL2     TRA         *+1                             TAKE CONTROL.
45            LXD         0,1                             GET SPILL INDICATOR CODE.
46            TXL         *+3,1,8                         IF NOT FDP, RETURN TO ROUTINE.
47            TXH         TOBIG,1,10                      SPILL IN ACC ALONE IS NOT SIGNIFICANT.
48            TXL         TOBIG,1,9                       MQ SPILL MEANS QUOTIENT IS OUT OF RANGE.
49            TRA*        0                               CONTINUE ROUTINE IF OK.
50  0010      XEC                                         ADDRESS SUPPLIED BY THE PROGRAM.
51            STQ         FPO+2                           IF CONTROL REMAINS HERE,
52            STO         FPO+1                           THEN PROGRAM IS FINISHED.
53            LRS         33                              GO INTO PRINT
54            STA         FPO                             ROUTINE AND
55            TSX         PRINT,4                         PRINT, THEN
56            HTR                                         STOP.
```

Figure 120. Floating-Point Trap

flexible technique. Note the use made of the decrement bits at location 0000 and a use of the address field of location 0000 in the routine SPIL2.

Line 1. Location 0000 is cleared to erase any data that may have been placed in it by a previous routine. The symbolic address SPILL is placed in the XEC instruction located at 0010. This causes the trap routine to return control to this program. NOTE: a postmortem print that prints location 0010 informs the programmer that his program has passed this point.

Line 16. The TXI instruction regains control of the program from the trap routine, and sets the scaling signal by placing a 1 in XRB. Recall that an MQ spill was not significant at this point; therefore, with the partial sum stored, the bit code in the decrement portion of location 0000 is checked. If position 17 has a 1 while positions 14, 15, and 16 have 0's, the summation is continued and XRB is set to zero (line 21).

Line 22. If scaling has already been in progress, a second spill is defined as an error.

Line 24. A new control address is placed at location 0010, because a spill in the following routines is to be treated in a different fashion. (Again, the address field of location 0010 can inform the programmer that this point of the program has been processed).

Line 35. Note that the accumulator overflow indicator is related to fixed-point operations only, when the FPT is on.

Line 44. The transfer instruction seizes control from the trap routine. The decision is made as follows:

1. If the spill occurred in the FSB instruction, the octal code in the decrement of location zero is less than 0010. Such a spill is not significant and a return to the routine is made by an indirectly addressed TRA instruction.
2. If the spill occurred in the accumulator alone (on FDP), the octal code in the decrement of location 0000 is not greater than 0012 and not less than or equal to 0011. Accumulator spill, alone, at the FDP instruction is not significant. With an MQ spill, the conditions just stated are not met; the program proceeds to TOBIG location (MQ overflow or underflow is defined as a TOBIG condition).

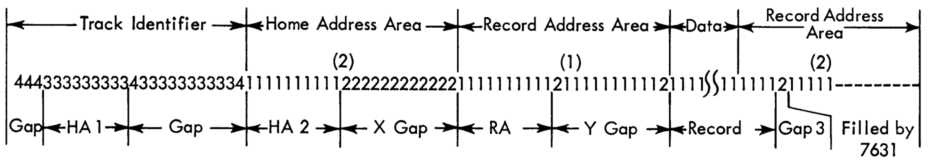
Writing a Format Track

Format track data organization in core storage is shown in Figure 121. The accompanying program listing shows all octal and symbolic notations with appropriate comments. The format track shown in Figure 121 is made up as follows:

1. Gap 1, home address 1 (HA1), and gap 2 are called the track identifier area and consist of 24 characters. They must be written with eight-bit mode characters. The remainder of the format track may be written with either six-bit or eight-bit mode characters, according to the requirements of the data track. Six-bit mode characters are used in Figure 121.

2. Home address 2 (HA2) and the X gap are called the home address area 2 and consist of 22 characters.

3. The record address (RA) and the Y gap are called the record address area 1 and consist of 22 characters. Record address area 2 consists of four characters of the data record (synchronization), a single character gap 3, and 11 characters after gap 3. The last 12 characters are automatically written by the 7631. The entire record address area equals 38 characters.



2880	Six-bit characters per track
-84	Identification and gaps
2796	Positions for data or,
466	Six-character words

Track Identifier	24 characters
Home Address Area 2	22 characters
Record Address Area (1 and 2)	38 characters
	84 characters

Figure 121. Format Track Core Storage Layout (Single Record)

Write and Write Check Format Track—Write, Write Check, and Read Single Record Operation

This program example is intended to show sample usage of instructions, commands, and orders and has not been tested on an operating system.

```
WRITE A FORMAT AT CYLINDER 0000, SINGLE RECORD PER TRACK.

                              CPU PROGRAM

00051  1 00000 0 00222        TCH CHECK       INTERRUPT TRANSFER
00100  0500 00 0 00317  START CLA TPAD1       SET UP TRAP
00101  0601 00 0 00021        STO 17          ADDRESS
00102  0564 00 0 00316        ENB ENCHD       ENABLE CHAN D
00103 -0541 00 0 00150        RSCD STFMT      START FORMAT

00104  2 00001 1 00104  PLAY  TIX *,1,1       KEEP CPU
00105  1 77776 1 00104        TXI *-1,1,-2    BUSY

00106  0000 00 0 00000        HTR **          NEVER STOP HERE

00107 -0641 00 0 00306  TRAP1 SCHD TEMP1      CHECK FOR
00110  0500 00 0 00306        CLA TEMP1       CORRECT
00111  0340 00 0 00307        CAS SCHD1       CHANNEL
00112  0020 00 0 00114        TRA *+2         STOP
00113  0020 00 0 00115        TRA *+2         OK

00114  0000 00 0 00114        HTR *           ERROR

00115  0500 00 0 00320        CLA TPAD2       SET UP NEW
00116  0601 00 0 00021        STO 17          TRAP ADDRESS
00117 -0541 00 0 00143        RSCD WRCKF      START WRITE CHECK
00120  0760 00 0 00014        RCT             RESTORE
00121  0020 00 0 00104        TRA PLAY        CPU BUSY
00122  0000 00 0 00000        HTR **          NEVER STOP HERE

00123  0000 00 0 00123  TRAP2 HTR *           CORRECT STOP

                             CHANNEL PROGRAM

WRITE FORMAT TRACK

00124  0000 01 2 00311  WRFMT XMT RSTRT,,1    POST RESTART ADDRESSES
00125  0 00000 0 00127        PZE WRFMT+3
00126 -2 40000 2 00055        LCC 45          SET UP FOR 460 WORDS
00127  2 40000 0 00257        CTLW DWRF       WRITE FORMAT CYL 0000

00130 -0000 04 0 00261  COPYS CPYP FTWD1,,4   TRACK IDENTIFICATION
00131 -0000 04 0 00265        CPYP FTWD2,,4   HA2,X GAP,2 CHAR OF RA
00132 -0000 01 0 00265        CPYP FTWD2,,1   RA
00133 -0000 01 0 00271        CPYP FTWD6,,1   RA,4 CHAR OF Y GAP
00134 -0000 01 0 00265        CPYP FTWD2,,1   Y GAP
00135 -0000 01 0 00272        CPYP FTWD7,,1   2 CHAR Y GAP,4 CHAR RCRD
00136 -0000 12 0 00273        CPYP FTWD8,,10  DATA RECORD AREA
00137 -2 40000 0 00136        TDC *-1         46 TIMES FOR 460 WORDS
00140 -0000 06 0 00273        CPYP FTWD8,,6   6 MORE DATA WDS FOR 466
00141 -1 00001 0 00305        CPYD FTWD9,,1   6 FOR 3 GAP

00142  3 40000 0 00142  WFEND TWT *

WRITE CHECK FORMAT TRACK

00143  0000 01 2 00311  WRCKF XMT RSTRT,,1    POST RESTART ADDRESS
00144  0 00000 0 00146        PZE WRCKF+3
00145 -2 40000 2 00055        LCC 45
00146  2 40000 0 00253        CTLW DWRC1
00147  1 00000 0 00130        TCH COPYS

00150  0000 01 2 00310  STFMT XMT WAIT,,1     POST RETURN ADDRESSES
00151  0 00000 0 00155        PZE STFMT+5
00152  0000 01 2 00311        XMT RSTRT,,1
00153  0 00000 0 00124        PZE WRFMT
00154  2 00000 0 00245        CTL DSEK1       SEEK ACC 0, MOD 1
00155  0000 00 0 00155        WTR *           WAIT FOR SEEK
```

Write, Write Check, and Read on Disk Storage

Using the format written with the first program example, the following program listing shows the main program steps necessary to write, write check, and then read a 466-word record (maximum) using access 0, module 1, track 0038, and record address 927601. Both this program listing and the previous one are intended to show sample usage of instructions, commands, and orders and neither program has actually been tested on an operating system.

```
                              CPU PROGRAM

00156  0564 00 0 00316   BEGIN ENB ENCHD         ENABLE TRAP
00157 -0541 00 0 00164         RSCD GO           START CHANNEL D

CPU IS NOW FREE TO PERFORM OTHER TASKS WHILE THE 7909 FINDS THE RECORD
AND PROCESSES IT.

00160  2 00001 1 00160         TIX *,1,1         WAIT FOR TRAP
00161  1 77776 1 00160         TXI *-1,1,-2

00162  0000 00 0 00162   DONE  HTR *             JOB COMPLETE

00163  0000 00 0 00163   HELP  HTR *             CPU STOPS HERE ON ERROR

                              CHANNEL PROGRAM

00164  0000 01 2 00311   GO    XMT RSTRT,,1      POST RESTART ADDRESS
00165  0 00000 0 00172         PZE WRITE
00166  0000 01 2 00310         XMT WAIT,,1
00167  0 00000 0 00171         PZE GO+5

SEEK DISK ADDRESS 0038 AND WAIT FOR ATTENTION SIGNAL

00170  2 00000 0 00247         CTL DSEK2         SEEK DISK ADDRESS 0038
00171  0000 00 0 00171         WTR *             WAIT FOR ATTENTION

00172 -2 40000 2 00011   WRITE LCC 9             NUMBER OF RETRIES
00173  0000 01 2 00311         XMT RSTRT,,1      RESTART ADDRESS
00174  0 00000 0 00175         PZE WRITE+3

WRITE 466 WORDS AT RECORD ADDRESS 927601

00175  2 40000 0 00251         CTLW DVSR         VERIFY SINGLE RECORD
00176 -0003 43 0 10000         CPYP DATA1,,227 WRITE 466 WORDS
00177 -0002 43 0 15000         CPYP DATA2,,163
00200 -1 00114 0 20000         CPYD DATA3,,76

00201  3 00000 0 00312         LAR BRANC         CHECK FOR WRITE CHECK DONE
00202 -1 60001 2 00212         TCM READ,,1,6     GO TO READ
00203  3 00000 0 00315         LAR ONE
00204  3 00000 2 00312         SAR BRANC

00205 -2 40000 6 00172   WRCHK LCC WRITE,4       RELOAD NUMBER OF RETRIES
00206  0000 01 2 00311         XMT RSTRT,,1      RESTART ADDRESS
00207  0 00000 0 00210         PZE WRCHK+3

WRITE CHECK RECORD AT RECORD ADDRESS 927601

00210  2 40000 0 00255         CTLW DWRC2
00211  1 00000 0 00176         TCH WRITE+4       DO WRITE CHECK

00212 -2 40000 6 00172   READ  LCC WRITE,4       RELOAD RETRIES
00213  0000 01 2 00311         XMT RSTRT,,1      RESTART ADDRESS
00214  0 00000 0 00215         PZE READ+3

READ RECORD AT RECORD ADDRESS 927601

00215  2 00000 2 00251         CTLR DVSR         VERIFY SINGLE RECORD
00216 -1 00722 0 25000         CPYD DATA4,,466 READ FULL RECORD

00217  0000 01 2 00021         XMT 17,,1
00220  0020 00 0 00162         TRA DONE
00221  3 40000 0 00221         TWT *             WRITE-WRITE CHECK-READ
                                                 COMPLETE - TRAP CPU FOR
                                                 PROCESSING
```

```
                         7909 INTERRUPT ROUTINE

00222 -1 00002 4 00224   CHECK TCM *+2,,100        WAS THIS ATTENTION 1
00223  1 00000 0 00232         TCH ERROR           NO
00224  2 40000 2 00000         SNS                 YES
00225 -1 00001 0 00313         CPYD SENSE,,1       GET FIRST SENSE WORD

00226  3 00000 0 00313         LAR SENSE
00227 -1 60004 2 00231         TCM *+2,,100,6      ACCESS 0, MODULE 1
00230  1 00000 6 00310         LIPT WAIT,4         WAIT FOR CORRECT ATTENTION
00231  1 00000 6 00311         LIPT RSTRT,4        START WRITE

00232 -2 40000 0 00237   ERROR TDC *+5             REDUCE TRIES
00233  0000 01 2 00021         XMT 17,,1           TEN TRIES DONE TRAP CPU
00234  0020 00 0 00163         TRA HELP            FOR HELP

00235  3 40000 0 00236         TWT *+1             PREPARE TO RETURN WITH STC

00236  1 00000 6 00311         LIPT RSTRT,4

00237 -1 00040 2 00244         TCM RTRN,,100000    CHECK FOR I-O CHECK
00240 -1 00020 2 00233         TCM ERROR+1,,10000  SEQU CHECK CALL CPU
00241 -1 00010 2 00244         TCM RTRN,,1000      UNUSUAL END
00242 -1 00001 2 00233         TCM ERROR+1,,1      ADAPTER CHECK CALL CPU
00243  1 00000 0 00233         TCH ERROR+1         POSSIBLE MULTIPLE CONDITION
                                                   CALL CPU

00244  1 00000 6 00311   RTRN  LIPT RSTRT,4        TRY AGAIN

                            7631 ORDERS

00245 +101212011212      DSEK1 OCT 101212011212 DSEK - 80
00246 +120100000000            OCT 120100000000 01-0001-00

00247 +101212011212      DSEK2 OCT 101212011212 DSEK - 80
00250 +031000000000            OCT 031000000000 01-0038-00

00251 +100212011102      DVSR  OCT 100212011102 DVSR - 82
00252 +070612010000            OCT 070612010000 01-9276-01

00253 +100612011212      DWRC1 OCT 100612011212 DWRC - 86
00254 +120100000000            OCT 120100000000 01-0001-00

00255 +100612011102      DWRC2 OCT 100612011102 DWRC - 86
00256 +070612010000            OCT 070612010000 01-9276-01

00257 +100312011212      DWRF  OCT 100312011212 DWRF - 83
00260 +120100000000            OCT 120100000000 01-0001-00

                           FORMAT AREAS

00261  040404030303      FTWD1 BCD 444333333333433333333334
00262  030303030303
00263  040303030303
00264  030303030304
00265  010101010101      FTWD2 BCD 1111111
00266  010101010202      FTWD3 BCD 1111122
00267  020202020202      FTWD4 BCD 1222222
00270  020202020101      FTWD5 BCD 1222211
00271  010102010101      FTWD6 BCD 1112111
00272  010201010101      FTWD7 BCD 1121111
00273  010101010101      FTWD8 BCD 511111111111111111111111111111
00274  010101010101
00275  010101010101
00276  010101010101
00277  010101010101
00300  010101010101            BCD 511111111111111111111111111111
00301  010101010101
00302  010101010101
00303  010101010101
00304  010101010101
00305  020101010101      FTWD9 BCD 1211111

                              CONSTANTS

00306  0 00000 0 00000   TEMP1 PZE
00307  0 00142 0 00142   SCHD1 PZE WFEND,,WFEND
00310  0 00000 0 00000   WAIT  PZE **              ATTENTION RETURN ADDRESS
00311  0 00000 0 00000   RSTRT PZE **              CHANNEL RETURN ADDRESS
00312  0 00000 0 00000   BRANC PZE **              WRITE CHECK CONTROL
                 00313   SENSE BSS 2               LOCATION FOR SENSE WORDS
00315 +000000000001      ONE   DEC 1
00316 +000000000010      ENCHD OCT 10
00317  0020 00 0 00107   TPAD1 TRA TRAP1
00320  0020 00 0 00123   TPAD2 TRA TRAP2
```

Appendix A

Number Systems and Conversion

The common decimal notation of the commercial and scientific world is familiar to all of us. This notation is so familiar that you probably have never before questioned its use. Could it be possible that, for some purposes, another system is more convenient? The decision 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 the octal system and would be questioning the decimal system.

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

Current digital computers use binary circuits and the mathematics of the computers is therefore binary in nature. The only convenient way to learn the operation of a computer is to learn the binary system. The octonary or 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.

Perhaps, as a first step, it would be well to see what is meant by the binary system of numbers. The binary, or base-two system, uses two symbols, 0 and 1, to represent all quantities. Counting is started in the binary system in the same manner as in the decimal system with 0 for zero and 1 for one. At two in the binary system it is found that there are no more symbols to be used. It is therefore necessary to take the same move at two in the binary system that is taken at ten in the decimal system. This move 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
0	0	101	5
1	1	110	6
10	2	111	7
11	3	1000	8
100	4	1001	9

The binary system is used in computers because all present components are inherently binary. That is, a relay maintains its contacts either closed or open, magnetic materials are utilized by magnetizing them in one direction or the other, a vacuum tube is conveniently maintained either fully conducting or nonconducting, or the transmission of information along a wire may be accomplished by transmitting or not transmitting an electrical pulse at a certain time.

Although binary numbers in general have more terms than their decimal counterparts (about 3.3 times as many), computation in the binary system is quite simple.

For *addition,* it is only necessary to remember the following three rules:

1. Zero plus zero equals zero.
2. Zero plus one equals one.
3. One plus one equals zero with a carry of one to the next position on the left. To see how the rules work, consider the addition of 15 plus 7 with these numbers expressed in binary notation:

	SIXTEENS	EIGHTS	FOURS	TWOS	ONES
(carries)	(1)	(1)	(1)	(1)	
	0	1	1	1	1 = 15
	+ 0	0	1	1	1 = 7
	1	0	1	1	0 = 22

In the ones column we have 1 plus 1 for a sum of 0 and a 1 carried to the two column. In the twos column we have 1 plus 1 for a sum of 0 but we must also add the carry from the ones column, making a final sum of 1 with a carry to the fours column. The same procedure occurs in the fours column. In the eights column we have a 1 plus a 0 giving a sum of 1, but adding in the carry from the fours column makes the final sum 0 with a carry to the sixteens column. In this column we have 0 plus 0 giving a sum of 0 and to this we add the carry from the eights column, making a final sum of 1.

The resultant sum of the addition contains 1's in the sixteens, fours, and twos columns, which is the binary representation of 22, the correct sum of 15 plus 7 (16 plus 4 plus 2 equals 22).

The rules for *subtraction* of binary digits are equally simple:

1. Zero minus zero equals zero.
2. One minus one equals zero.

3. One minus zero equals one.
4. Zero minus one equals one, with one borrowed from the left.

Using the same numbers as we did in the addition, the subtraction works as follows:

	SIXTEENS	EIGHTS	FOURS	TWOS	ONES	
(borrows)	0	0	0	0	0	
	0	1	1	1	1	= 15
	− 0	0	1	1	1	= 7
	0	1	0	0	0	= 8

In the ones column we have 1 minus 1 for a sum of 0 with no borrows. The same procedure occurs in the twos and fours columns. In the eights column we have 1 minus 0 for a sum of 1. In the sixteens column we have 0 minus 0 for a sum of 0. With the subtraction finished we have 1's in the eights column only, signifying the answer to be 8.

For *multiplication* only three rules need to be remembered:

1. Zero times zero equals zero.
2. Zero times one equals zero; no carries are considered.
3. One times one equals one.

The binary multiplication table is such that all that is necessary when multiplying one number (multiplicand) by another (multiplier) is to examine the multiplier digits one at a time and, each time a 1 is found, add the multiplicand into the result, and each time a 0 is found add nothing. Of course, the multiplicand must be shifted for each multiplier digit, but this is not different from the shifting that is done in the decimal system.

An example of binary multiplication is 26 multiplied by 19:

```
DECIMAL                                              BINARY
  26 = 16 + 8 + 0 + 2 + 0          =         11010
× 19 = 16 + 0 + 0 + 2 + 1          =         10011
   Using the above rules, the product        11010
   will be arrived at by a series           11010
   of adding the multiplicand              00000
   and shifting whenever                  00000
   a 1 is found in the                   11010
   multiplier.                           111101110
```

Interpreting the binary result of the multiplication by using the ones, twos, fours, . . . etc., system we find that we have,

256 + 128 + 64 + 32 + 0 + 8 + 4 + 2 + 0

which equals 494, thus proving the problem.

Binary division is accomplished by applying similar concepts. From the examples of addition, subtraction, and multiplication, it may be seen that whatever operation the computer is working on will be accomplished by repetitive addition.

The computer operates internally using the binary system. However, it is able to convert from one system to another by use of a stored program. Thus, input-output data may be expressed in decimal (or any other) form when the operator finds it more convenient to do so.

Octal Number System

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 much of a problem to the computer itself. However, 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 or radix 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 in this number system. The important relationship to remember is that three binary positions are equivalent to one octal position. The following table is used constantly when working on or about the computer.

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

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

BINARY	OCTAL
001 000	10
001 001	11
001 010	12
001 011	13
001 100	14

and so on.

Remember that 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.

Number Conversions

Before an attempt is made to convert numbers from one system to another, it is best to review what a number represents. In the demical system a number is represented or expressed by a sum of terms. Each individual term consists of a product of a power of ten and some integer from 0 to 9. For example, the number 123 means 100 plus 20 plus 3. This may also be expressed as:

$$(1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$$

Ten is said to be the base or radix of this system because of the role that the powers of 10 and the integers up to 10 play in the above expansion. If two is chosen as the base, numbers are said to be represented in the binary system. Consider the binary number 1 111 011. What do these zeros and ones represent? They represent the coefficients of the ascending powers of 2. Expressed in another way the number is:

$$(1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) +$$
$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

The various orders do not have the meaning of units, tens, hundreds, thousands, etc., as in the decimal system; instead they signify units, twos, fours, eights, sixteens, etc. In applying the above information it is found that the number 123 breaks down in both systems as follows:

```
BINARY                                  DECIMAL
1  111  011                             1 2 3
|  |||  ||└─1  units                    | | └─3  units
|  |||  |└──2  twos                     | └──20  tens
|  |||  └───0  fours                    └───100  hundreds
|  ||└──────8  eights                       ───
|  |└──────16  sixteens                     123
|  └───────32  thirty-twos
└──────────64  sixty-fours
           ───
           123
```

In the octal system, a number is represented in the same manner except that the base is 8. The digits of the number represent the coefficients of the ascending powers of 8. Consider the octal number:

$$\begin{aligned} 173 &= (1 \times 8^2) + (7 \times 8^1) + (3 \times 8^0) \\ &= \quad 64 \quad + \quad 56 \quad + \quad 3 \\ &= 123 \text{ (decimal)} \end{aligned}$$

Similarly:

```
Octal 173
      ||└─3  units
      |└─56  eights
      └──64  sixty-fours
```

By remembering what a number represents in the binary or octal system, the number can be converted to its decimal equivalent by the method shown above. As the numbers get bigger, this method becomes quite impossible to use. The following section provides detailed methods for converting from one system to another.

Integers

DECIMAL TO OCTAL

Convert the decimal number 149 to its octal equivalent. RULE: Divide the decimal number by 8 and develop the octal number as per example.

```
8 |_149  Remainder 5   ↑
8   |_18    "      2   |   = 225
8     |_2   "      2   |
        0            read
```

We first divided the original number to be converted by 8. The remainder of this first division becomes the low-order digit of the conversion (5). We then divide the quotient (received from the first division) by 8. Again the remainder becomes a part of the answer (next higher order, 2). This is continued until the quotient is smaller than the divisor. At this time the final quotient is considered the high order of the conversion (2).

OCTAL TO DECIMAL

Convert the octal number 225 to its decimal equivalent. RULE: Multiply by 8 and add, as per example.

```
  2 2 5
 × 8 | |
 ───  | |
  16  | |
 + 2←┘ |
 ───    |
  18    |
 × 8    |
 ───    |
 144    |
 + 5←──┘
 ───
 149
```

The high-order digit is multiplied by 8 and the next lower-order digit is added to the result. The resultant answer is then multiplied by 8 and the next lower-order digit is added to the result. When the low-order digit has been added to the answer, the process ends. In the following examples, where multiplication or division is used, detailed explanations will not be used because the operations are similar.

Octal to Binary and Binary to Octal

Rule: Express the number in binary groups of three.

OCTAL TO BINARY

$\overbrace{010}^{2}\ \overbrace{010}^{2}\ \overbrace{101}^{5} = 010\ \ 010\ \ 101$

BINARY TO OCTAL

$\underbrace{010}_{2}\ \underbrace{010}_{2}\ \underbrace{101}_{5} = 225$

Decimal to Binary

Rule: Divide the decimal number by 2 and develop as per example; convert 149 to its binary equivalent.

```
2|_149  Remainder  1 ↑
2|_74      "       0 |
2 |_37     "       1 |
2 |_18     "       0 |
2 |_9      "       1 |  = 010  010  101
2 |_4      "       0 |
2 |_2      "       0 |
2 |_1      "       1 |
    0      "      read
```

Binary to Decimal

Rule: Multiply by 2 and add as per example; convert 010 010 101 to its decimal equivalent.

```
 10  010  101
×  2
   2
+  0
   2
×  2
   4
+  0 ←
   4
×  2
   8
+  1 ←
   9
×  2
  18
+  0 ←
  18
×  2
  36
+  1 ←
  37
×  2
  74
+  0 ←
  74
×  2
 148
+  1 ←
 149
```

OR 10 010 101

$= 1(2^7) + 0(2^6) + 0(2^5) + 1(2^4) + 0(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$

$= 128 + 16 + 4 + 1$

$= 149$

Fractions

Decimal to Octal

Rule: Multiply by 8 and develop the octal number as per example:

```
Read    .149
 |   •  × 8
 |   1 | .192
 |     | × 8
 |   1 | .536
 |     | × 8
 |   4 | .288
 |     | × 8
 ↓   2 | .304
       =  .1142 +
```

Octal to Decimal

Rule: Express as powers of 8, add and divide as per example:

$.1142 = 1(8^{-1}) + 1(8^{-2}) + 4(8^{-3}) + 2(8^{-4})$
$= 1/8 + 1/64 + 4/512 + 2/4096$
$= 610/4096$
$= .1489$ plus
or .149

Octal to Binary and Binary to Octal

Rule: The same rule applies for fractions as for whole numbers.

Example:

$\overbrace{.001}^{.1}\ \overbrace{001}^{1}\ \overbrace{100}^{4}\ \overbrace{010}^{2}$ $\quad$ $\underbrace{.001}_{.1}\ \underbrace{001}_{1}\ \underbrace{100}_{4}\ \underbrace{010}_{2}$

Binary to Decimal

The same rule applies as for whole numbers; for example:

.001 001 100 010

$= 1(2^{-3}) + 1(2^{-6}) + 1(2^{-7}) + 1(2^{-11})$
$= 1/8 + 1/64 + 1/128 + 1/2048$
$= 305/2048$
$= .1489$ plus
or .149

Decimal to Binary

The same rule applies as for whole numbers. For example:

```
Read   .149
 •     × 2
 0    .298
      × 2
 0    .596
      × 2
 1    .192
      × 2
 0    .384
      × 2
 0    .768
      × 2
 1    .536
      × 2
 1    .072
      × 2
 0    .144
      × 2
 0    .288
      × 2
 0    .576
      × 2
 1    .152
      × 2
 0    .304
 =    .001 001 100 010 +
```

Improper Fractions

Decimal to Binary

This requires conversion from decimal to octal and then to binary. For example, convert 149.149 to its binary equivalent.

```
8 |149.  remainder 5  ↑              .149
8 |18.       "     2  |             × 8
8 |2.        "     2  |         1   .192
   0                 read.          × 8
                                1   .536
                                    × 8
                                4   .288
                                    × 8
                          read. 2   .304
```

$$= \underbrace{010}_{2}\ \underbrace{010}_{2}\ \underbrace{101}_{5}\bullet\underbrace{001}_{1}\ \underbrace{001}_{1}\ \underbrace{100}_{4}\ \underbrace{010}_{2}$$

$$149.149_{10} = 225.1142_8 = 010\ 010\ 101.001\ 001\ 100\ 010_2$$

Binary to Decimal

This requires conversion from binary to octal and then to decimal.

Convert to decimal:

$$\underbrace{010}_{2}\ \underbrace{010}_{2}\ \underbrace{101}_{5}\bullet\underbrace{001}_{1}\ \underbrace{001}_{1}\ \underbrace{100}_{4}\ \underbrace{010}_{2}$$

```
= 2   2   5 • 1   1   4   2
 × 8
  16
 + 2
  18
 × 8
 144
 + 5
 149.
```

$$\frac{1}{8} + \frac{1}{64} + \frac{4}{512} + \frac{2}{4096} = \frac{610}{4096} = .149$$

As with decimal-to-binary, conversion of the integer and fraction parts is performed independently.

Floating-Point Word

Decimal to Floating Point

Convert decimal 149.149 to normal floating-point word.

Decimal to octal:

$149.149_{10} = 225.1142_8$

Octal to binary:

$225.1142_8 = 010\ 010\ \ 101.001\ 001\ 100\ 010_2$

Binary to floating point word:

$10\ 010\ 101.001\ 001\ 100\ 010 \times 2^0 =$

$.10\ 010\ 101\ 001\ 001\ 100\ 010 \times 2^8$

$8 + 128 = 136$ (Characteristic)

$$\underbrace{10\ 001\ 000}_{\text{Characteristic}}.\underbrace{100\ 101\ 010\ 010\ 011\ 000\ 1}_{\text{Fraction}}\ \text{FP}$$

$2\ 1\ 0\ .\ 4\ 5\ 2\ 2\ 3\ 0\ 4_8$

NOTE: Word is normal if the fraction is less than 1, but greater than or equal to one-half.

Appendix B. Octal-Decimal Integer Conversion Table

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

Octal-Decimal Integer Conversion Table

4000	2048
to	to
4777	2559
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000	2560
to	to
5777	3071
(Octal)	(Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000	3072
to	to
6777	3583
(Octal)	(Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000	3584
to	to
7777	4095
(Octal)	(Decimal)

Appendix C. Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversion Table

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

Appendix D

Table of Powers of Two

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

Appendix E. SCAT Mnemonic Operation Codes

CODE	COMMENT	INDEXABLE	IND. ADDRESS	PAGE
ACL	Add and Carry Logical Word	X	X	21
ADD	Add	X	X	20
ADM	Add Magnitude	X	X	20
ALS	Accumulator Left Shift	X		31
ANA	AND to Accumulator	X	X	48
ANS	AND to Storage	X	X	48
ARS	Accumulator Right Shift	X		32
AXC	Address to Index, Complemented			46
AXT	Address to Index, True			45
BSFA	Backspace File, Ch. A	X		58
BSFB	Backspace File, Ch. B	X		58
BSFC	Backspace File, Ch. C	X		58
BSFD	Backspace File, Ch. D	X		58
BSFE	Backspace File, Ch. E	X		58
BSFF	Backspace File, Ch. F	X		58
BSFG*	Backspace File, Ch. G	X		58
BSFH*	Backspace File, Ch. H	X		58
BSR	Backspace Record	X		58
BSRA	Backspace Record, Ch. A	X		58
BSRB	Backspace Record, Ch. B	X		58
BSRC	Backspace Record, Ch. C	X		58
BSRD	Backspace Record, Ch. D	X		58
BSRE	Backspace Record, Ch. E	X		58
BSRF	Backspace Record, Ch. F	X		58
BSRG*	Backspace Record, Ch. G	X		58
BSRH*	Backspace Record, Ch. H	X		58
BTTA	Beginning of Tape Test, Ch. A	X		41
BTTB	Beginning of Tape Test, Ch. B	X		41
BTTC	Beginning of Tape Test, Ch. C	X		41
BTTD	Beginning of Tape Test, Ch. D	X		41
BTTE	Beginning of Tape Test, Ch. E	X		41
BTTF	Beginning of Tape Test, Ch. F	X		41
BTTG*	Beginning of Tape Test, Ch. G	X		41
BTTH*	Beginning of Tape Test, Ch. H	X		41
CAD**	Copy and Add Logical Word	X		
CAL	Clear and Add Logical Word	X	X	20
CAQ	Convert by Addition from MQ			57
CAS	Compare Accumulator with Storage	X	X	43
CFF	Change Film Frame	X		40
CHS	Change Sign	X		49
CLA	Clear and Add	X	X	20
CLM	Clear Magnitude	X		49
CLS	Clear and Subtract	X	X	20
COM	Complement Magnitude	X		49
CPY**	Copy	X		
CRQ	Convert by Replacement from MQ			56
CVR	Convert by Replacement from AC			56
DCT	Divide Check Test	X		42
DVH	Divide or Halt	X	X	24
DVP	Divide or Proceed	X	X	24
ECTM	Enter Copy Trap Mode	X		66
EFTM	Enter Floating Trap Mode	X		66
ENB	Enable from Y	X	X	65
ENK	Enter Keys	X		35
ERA	Exclusive OR to Accumulator	X	X	49
ESNT	Enter Storage Nullification, and Transfer	X	X	65
ESTM	Enter Select Trap Mode	X		66
ETM	Enter Trapping Mode	X		36
ETTA	End of Tape Test, Ch. A	X		41
ETTB	End of Tape Test, Ch. B	X		41
ETTC	End of Tape Test, Ch. C	X		41
ETTD	End of Tape Test, Ch. D	X		41
ETTE	End of Tape Test, Ch. E	X		41
ETTF	End of Tape Test, Ch. F	X		41
ETTG*	End of Tape Test, Ch. G	X		41
ETTH*	End of Tape Test, Ch. H	X		41
FAD	Floating Add	X	X	26
FAM	Floating Add Magnitude	X	X	27
FDH	Floating Divide or Halt	X	X	30
FDP	Floating Divide or Proceed	X	X	30
FMP	Floating Multiply	X	X	29
FOR	Four			(1)
FRN	Floating Round	X		28
FSB	Floating Subtract	X	X	27
FSM	Floating Subtract Magnitude	X	X	28
FVE	Five			(1)
HPR	Halt and Proceed			35
HTR	Halt and Transfer	X	X	36
IIA	Invert Indicators from Accumulator			53
IIL	Invert Indicators of the Left Half			53
IIR	Invert Indicators of the Right Half			53
IIS	Invert Indicators from Storage	X	X	53
IOCD	Input-Output under Count Control and Disconnect			62
IOCDN	(IOCD with No Transmission)			62
IOCP	Input-Output under Count Control and Proceed			62
IOCPN	(IOCP with No Transmission)			62
IOCT	Input-Output under Count Control and Transfer			63
IOCTN	(IOCT with No Transmission)			63
IORP	Input-Output of a Record and Proceed			62
IORPN	(IORP with No Transmission)			62
IORT	Input-Output of a Record and Transfer			63
IORTN	(IORT with No Transmission)			63
IOSP	Input-Output until Signal then Proceed			63
IOSPN	(IOSP with No Transmission)			63
IOST	Input-Output until Signal then Transfer			64
IOSTN	(IOST with No Transmission)			64
IOT	Input-Output Check Test	X		42
LAC	Load Complement of Address in Index			45
LAS	Logical Compare Accumulator with Storage	X	X	43
LBT	Low-Order Bit Test	X		42
LCHA	Load Channel A	X	X	61
LCHB	Load Channel B	X	X	61
LCHC	Load Channel C	X	X	61
LCHD	Load Channel D	X	X	61
LCHE	Load Channel E	X	X	61
LCHF	Load Channel F	X	X	61
LCHG*	Load Channel G	X	X	61
LCHH*	Load Channel H	X	X	61
LDA**	Locate Drum Address	X	X	
LDC	Load Complement of Decrement in XR			45
LDI	Load Indicators	X	X	51
LDQ	Load the MQ	X	X	33
LFT	Left Half Indicators, Off Test			55
LFTM	Leave Floating Trap Mode	X		66
LGL	Logical Left Shift	X		32
LGR	Logical Right Shift	X		32
LLS	Long Left Shift	X		32
LNT	Left Half Indicators, On Test			54
LRS	Long Right Shift	X		32
LSNM	Leave Storage Nullification Mode	X		65
LTM	Leave Trapping Mode	X		37
LXA	Load Index from Address			45
LXD	Load Index from Decrement			45
MON	Minus One			(1)
MPR	Multiply and Round	X	X	22
MPY	Multiply	X	X	22
MSE	Minus Sense	X		41
MTH	Minus Three			(1)
MTW	Minus Two			(1)
MZE	Minus Zero			(1)
NOP	No Operation			35
NZT	Storage Not-Zero Test	X	X	43

(1) The mnemonic code is an extended code; no particular machine code is concerned with it.

* 7090 Instruction only.

** 709 Instruction only, not included in this manual.

SCAT Mnemonic Operation Codes *(Cont'd)*

CODE	COMMENT	INDEXABLE	IND. ADDRESS	PAGE
OAI	OR Accumulator to Indicators			51
OFT	Off Test for Indicators	X	X	54
ONT	On Test for Indicators	X	X	54
ORA	OR to Accumulator	X	X	48
ORS	OR to Storage	X	X	48
OSI	OR Storage to Indicators	X	X	51
PAC	Place Complement of Address in XR			46
PAI	Place Accumulator in Indicators			50
PAX	Place Address in Index			46
PBT	P-Bit Test	X		42
PDC	Place Complement of Decrement in XR			46
PDX	Place Decrement in Index			46
PIA	Place Indicators in Accumulator			51
PON	Plus One			(1)
PSE	Plus Sense	X		40
PTH	Plus Three			(1)
PTW	Plus Two			(1)
PXA	Place Index in Address			47
PXD	Place Index in Decrement			47
PZE	Plus Zero			(1)
RCDA	Read Card Reader, Ch. A	X		58
RCDB*	Read Card Reader, Ch. B	X		58
RCDC	Read Card Reader, Ch. C	X		58
RCDD*	Read Card Reader, Ch. D	X		58
RCDE	Read Card Reader, Ch. E	X		58
RCDF*	Read Card Reader, Ch. F	X		58
RCDG*	Read Card Reader, Ch. G	X		58
RCDH*	Read Card Reader, Ch. H	X		58
RCHA	Reset and Load, Ch. A	X	X	60
RCHB	Reset and Load, Ch. B	X	X	60
RCHC	Reset and Load, Ch. C	X	X	60
RCHD	Reset and Load, Ch. D	X	X	60
RCHE	Reset and Load, Ch. E	X	X	60
RCHF	Reset and Load, Ch. F	X	X	60
RCHG*	Reset and Load, Ch. G	X	X	60
RCHH*	Reset and Load, Ch. H	X	X	60
RCT	Restore Channel Traps	X		64
RDCA	Reset Data Channel A	X		59
RDCB	Reset Data Channel B	X		59
RDCC	Reset Data Channel C	X		59
RDCD	Reset Data Channel D	X		59
RDCE	Reset Data Channel E	X		59
RDCF	Reset Data Channel F	X		59
RDCG	Reset Data Channel G	X		59
RDCH	Reset Data Channel H	X		59
RDR**	Read Drum	X		
RDS	Read Select	X		58
REWA	Rewind, Ch. A	X		59
REWB	Rewind, Ch. B	X		59
REWC	Rewind, Ch. C	X		59
REWD	Rewind, Ch. D	X		59
REWE	Rewind, Ch. E	X		59
REWF	Rewind, Ch. F	X		59
REWG*	Rewind, Ch. G	X		59
REWH*	Rewind, Ch. H	X		59
RFT	Right Half Indicators, Off Test			55
RIA	Reset Indicators from Accumulator			52
RIL	Reset Indicators of Left Half			52
RIR	Reset Indicators of Right Half			52
RIS	Reset Indicators from Storage	X	X	52
RND	Round	X		22
RNT	Right Half Indicators, On Test			54
RPRA	Read Printer, Ch. A	X		58
RPRB*	Read Printer, Ch. B	X		58
RPRC	Read Printer, Ch. C	X		58
RPRD*	Read Printer, Ch. D	X		58
RPRE	Read Printer, Ch. E	X		58
RPRF*	Read Printer, Ch. F	X		58
RPRG*	Read Printer, Ch. G	X		58
RPRH*	Read Printer, Ch. H	X		58
RQL	Rotate MQ Left	X		32
RTBA	Read Tape Binary, Ch. A	X		58
RTBB	Read Tape Binary, Ch. B	X		58
RTBC	Read Tape Binary, Ch. C	X		58
RTBD	Read Tape Binary, Ch. D	X		58
RTBE	Read Tape Binary, Ch. E	X		58
RTBF	Read Tape Binary, Ch. F	X		58
RTBG*	Read Tape Binary, Ch. G	X		58
RTBH*	Read Tape Binary, Ch. H	X		58
RTDA	Read Tape Decimal, Ch. A	X		58
RTDB	Read Tape Decimal, Ch. B	X		58
RTDC	Read Tape Decimal, Ch. C	X		58
RTDD	Read Tape Decimal, Ch. D	X		58
RTDE	Read Tape Decimal, Ch. E	X		58
RTDF	Read Tape Decimal, Ch. F	X		58
RTDG*	Read Tape Decimal, Ch. G	X		58
RTDH*	Read Tape Decimal, Ch. H	X		58
RUNA	Rewind and Unload Channel A	X		59
RUNB	Rewind and Unload Channel B	X		59
RUNC	Rewind and Unload Channel C	X		59
RUND	Rewind and Unload Channel D	X		59
RUNE	Rewind and Unload Channel E	X		59
RUNF	Rewind and Unload Channel F	X		59
RUNG*	Rewind and Unload Channel G	X		59
RUNH*	Rewind and Unload Channel H	X		59
SBM	Subtract Magnitude	X	X	21
SCHA	Store, Ch. A	X	X	60
SCHB	Store, Ch. B	X	X	60
SCHC	Store, Ch. C	X	X	60
SCHD	Store, Ch. D	X	X	60
SCHE	Store, Ch. E	X	X	60
SCHF	Store, Ch. F	X	X	60
SCHG*	Store, Ch. G	X	X	60
SCHH*	Store, Ch. H	X	X	60
SDLA	Set Density Low, Channel A			59
SDLB	Set Density Low, Channel B			59
SDLC	Set Density Low, Channel C			59
SDLD	Set Density Low, Channel D			59
SDLE	Set Density Low, Channel E			59
SDLF	Set Density Low, Channel F			59
SDLG*	Set Density Low, Channel G			59
SDLH*	Set Density Low, Channel H			59
SDHA	Set Density High, Channel A			59
SDHB	Set Density High, Channel B			59
SDHC	Set Density High, Channel C			59
SDHD	Set Density High, Channel D			59
SDHE	Set Density High, Channel E			59
SDHF	Set Density High, Channel F			59
SDHG*	Set Density High, Channel G			59
SDHH*	Set Density High, Channel H			59
SIL	Set Indicators of Left Half			51
SIR	Set Indicators of Right Half			52
SIX	Six			(1)
SLF	Sense Lights Off	X		40
SLN	Sense Lights On	X		40
SLQ	Store Left Half MQ	X	X	33
SLT	Sense Light Test	X		40
SLW	Store Logical Word	X	X	33
SPRA	Sense Printer, Ch. A	X		40
SPRB*	Sense Printer, Ch. B	X		40
SPRC	Sense Printer, Ch. C	X		40
SPRD*	Sense Printer, Ch. D	X		40
SPRE	Sense Printer, Ch. E	X		40
SPRF*	Sense Printer, Ch. F	X		40
SPRG*	Sense Printer, Ch. G	X		40
SPRH*	Sense Printer, Ch. H	X		40
SPTA	Sense Printer Test, Ch. A	X		40
SPTB*	Sense Printer Test, Ch. B	X		40
SPTC	Sense Printer Test, Ch. C	X		40
SPTD*	Sense Printer Test, Ch. D	X		40
SPTE	Sense Printer Test, Ch. E	X		40

(1) The mnemonic code is an extended code; no particular machine code is concerned with it.

* 7090 Instruction only.

** 709 Instruction only, not included in this manual.

SCAT Mnemonic Operation Codes *(Cont'd)*

CODE	COMMENT	INDEXABLE	IND. ADDRESS	PAGE
SPTF*	Sense Printer Test, Ch. F	X		40
SPTG*	Sense Printer Test, Ch. G	X		40
SPTH*	Sense Printer Test, Ch. H	X		40
SPUA	Sense Punch, Ch. A	X		40
SPUB*	Sense Punch, Ch. B	X		40
SPUC	Sense Punch, Ch. C	X		40
SPUD*	Sense Punch, Ch. D	X		40
SPUE	Sense Punch, Ch. E	X		40
SPUF*	Sense Punch, Ch. F	X		40
SPUG*	Sense Punch, Ch. G	X		40
SPUH*	Sense Punch, Ch. H	X		40
SSM	Set Sign Minus	X		50
SSP	Set Sign Plus	X		50
STA	Store Address	X	X	34
STD	Store Decrement	X	X	34
STI	Store Indicators	X	X	51
STL	Store Instruction Location Counter	X	X	34
STO	Store	X	X	33
STP	Store Prefix	X	X	33
STQ	Store MQ	X	X	33
STR	Store Location and Trap			34
STT	Store Tag	X	X	34
STZ	Store Zero	X	X	34
SUB	Subtract	X	X	21
SVN	Seven			(1)
SWT	Sense Switch Test	X		40
SXA	Store Index in Address			46
SXD	Store Index in Decrement			46
TCH	Transfer in Channel			44
TCNA	Transfer on Ch. A Not in Operation	X	X	44
TCNB	Transfer on Ch. B Not in Operation	X	X	44
TCNC	Transfer on Ch. C Not in Operation	X	X	44
TCND	Transfer on Ch. D Not in Operation	X	X	44
TCNE	Transfer on Ch. E Not in Operation	X	X	44
TCNF	Transfer on Ch. F Not in Operation	X	X	44
TCNG*	Transfer on Ch. G Not in Operation	X	X	44
TCNH*	Transfer on Ch. H Not in Operation	X	X	44
TCOA	Transfer on Ch. A in Operation	X	X	43
TCOB	Transfer on Ch. B in Operation	X	X	43
TCOC	Transfer on Ch. C in Operation	X	X	43
TCOD	Transfer on Ch. D in Operation	X	X	43
TCOE	Transfer on Ch. E in Operation	X	X	43
TCOF	Transfer on Ch. F in Operation	X	X	43
TCOG*	Transfer on Ch. G in Operation	X	X	43
TCOH*	Transfer on Ch. H in Operation	X	X	43
TEFA	Transfer on End of File, Ch. A	X	X	44
TEFB	Transfer on End of File, Ch. B	X	X	44
TEFC	Transfer on End of File, Ch. C	X	X	44
TEFD	Transfer on End of File, Ch. D	X	X	44
TEFE	Transfer on End of File, Ch. E	X	X	44
TEFF	Transfer on End of File, Ch. F	X	X	44
TEFG*	Transfer on End of File, Ch. G	X	X	44
TEFH*	Transfer on End of File, Ch. H	X	X	44
TIF	Transfer if Indicators Off	X	X	53
TIO	Transfer if Indicators On	X	X	53
TIX	Transfer on Index			40
TLQ	Transfer on Low MQ	X	X	39
TMI	Transfer on Minus	X	X	38
TNO	Transfer on No Overflow	X	X	38
TNX	Transfer on No Index			40
TNZ	Transfer on No Zero	X	X	37
TOV	Transfer on Overflow	X	X	38
TPL	Transfer on Plus	X	X	38
TQO	Transfer on Quotient Overflow	X	X	38
TQP	Transfer on MQ Plus	X	X	38
TRA	Transfer	X	X	36
TRCA	Transfer on Redun. Check, Ch. A	X	X	44
TRCB	Transfer on Redun. Check, Ch. B	X	X	44
TRCC	Transfer on Redun. Check, Ch. C	X	X	44
TRCD	Transfer on Redun. Check, Ch. D	X	X	44
TRCE	Transfer on Redun. Check, Ch. E	X	X	44
TRCF	Transfer on Redun. Check, Ch. F	X	X	44
TRCG*	Transfer on Redun. Check, Ch. G	X	X	44
TRCH*	Transfer on Redun. Check, Ch. H	X	X	44
TSX	Transfer and Set Index			39
TTR	Trap Transfer	X	X	37
TXH	Transfer on Index High			39
TXI	Transfer with Index Incremented			39
TXL	Transfer on Index Low or Equal			40
TZE	Transfer on Zero	X	X	37
UAM	Unnormalized Add Magnitude	X	X	28
UFA	Unnormalized Floating Add	X	X	27
UFM	Unnormalized Floating Multiply	X	X	29
UFS	Unnormalized Floating Subtract	X	X	28
USM	Unnormalized Subtract Magnitude	X	X	28
VDH	Variable Length Divide or Halt	X		24
VDP	Variable Length Divide or Proceed	X		24
VLM	Variable Length Multiply	X		22
WDR**	Write Drum	X		
WEF	Write End of File	X		59
WEFA	Write End of File, Ch. A	X		59
WEFB	Write End of File, Ch. B	X		59
WEFC	Write End of File, Ch. C	X		59
WEFD	Write End of File, Ch. D	X		59
WEFE	Write End of File, Ch. E	X		59
WEFF	Write End of File, Ch. F	X		59
WEFG*	Write End of File, Ch. G	X		59
WEFH*	Write End of File, Ch. H	X		59
WPBA	Write Printer Binary, Ch. A	X		58
WPBB*	Write Printer Binary, Ch. B	X		58
WPBC	Write Printer Binary, Ch. C	X		58
WPBD*	Write Printer Binary, Ch. D	X		58
WPBE	Write Printer Binary, Ch. E	X		58
WPBF*	Write Printer Binary, Ch. F	X		58
WPBG*	Write Printer Binary, Ch. G	X		58
WPBH*	Write Printer Binary, Ch. H	X		58
WPDA	Write Printer Decimal, Ch. A	X		58
WPDB*	Write Printer Decimal, Ch. B	X		58
WPDC	Write Printer Decimal, Ch. C	X		58
WPDD*	Write Printer Decimal, Ch. D	X		58
WPDE	Write Printer Decimal, Ch. E	X		58
WPDF*	Write Printer Decimal, Ch. F	X		58
WPDG*	Write Printer Decimal, Ch. G	X		58
WPDH*	Write Printer Decimal, Ch. H	X		58
WPUA	Write Punch, Ch. A	X		58
WPUB*	Write Punch, Ch. B	X		58
WPUC	Write Punch, Ch. C	X		58
WPUD*	Write Punch, Ch. D	X		58
WPUE	Write Punch, Ch. E	X		58
WPUF*	Write Punch, Ch. F	X		58
WPUG*	Write Punch, Ch. G	X		58
WPUH*	Write Punch, Ch. H	X		58
WRS	Write Select	X		58
WTBA	Write Tape Binary, Ch. A	X		58
WTBB	Write Tape Binary, Ch. B	X		58
WTBC	Write Tape Binary, Ch. C	X		58
WTBD	Write Tape Binary, Ch. D	X		58
WTBE	Write Tape Binary, Ch. E	X		58
WTBF	Write Tape Binary, Ch. F	X		58
WTBG*	Write Tape Binary, Ch. G	X		58
WTBH*	Write Tape Binary, Ch. H	X		58
WTDA	Write Tape Decimal, Ch. A	X		58
WTDB	Write Tape Decimal, Ch. B	X		58
WTDC	Write Tape Decimal, Ch. C	X		58
WTDD	Write Tape Decimal, Ch. D	X		58
WTDE	Write Tape Decimal, Ch. E	X		58
WTDF	Write Tape Decimal, Ch. F	X		58
WTDG*	Write Tape Decimal, Ch. G	X		58
WTDH*	Write Tape Decimal, Ch. H	X		58
WTV**	Write Cathode Ray Tube	X		
XCA	Exchange Accumulator and MQ			34
XCL	Exchange Logical Accumulator and MQ			34
XEC	Execute	X	X	36
ZET	Storage Zero Test	X	X	43

(1) The mnemonic code is an extended code; no particular machine code is concerned with it.

* 7090 Instruction only.

** 709 Instruction only, not included in this manual.

Appendix F. Listing of Instructions

Alphabetic Listing

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
ACL	0361	Add and Carry Logical Word		X	X	21
ADD	0400	Add		X	X	20
ADM	0401	Add Magnitude		X	X	20
ALS	0767	Accumulator Left Shift	7	X		31
ANA	—0320	AND to Accumulator		X	X	48
ANS	0320	AND to Storage		X	X	48
ARS	0771	Accumulator Right Shift	7	X		32
AXC	—0774	Address to Index Complemented				46
AXT	0774	Address to Index True				45
BSF	—0764	Backspace File	8	X		58
BSR	0764	Backspace Record	8	X		58
BTT	0760..xxxx	Beginning of Tape Test		X		41
CAL	—0500	Clear and Add Logical Word		X	X	20
CAQ	—0114	Convert by Addition from MQ	6			57
CAS	0340	Compare AC with storage		X	X	43
CHS	0760..0002	Change Sign		X		49
CLA	0500	Clear and Add		X	X	20
CLM	0760..0000	Clear Magnitude		X		49
CLS	0502	Clear and Subtract		X	X	20
COM	0760..0006	Complement Magnitude		X		49
CRQ	—0154	Convert by Replacement from MQ	6			56
CVR	0114	Convert by Replacement from AC	6			56
DCT	0760..0012	Divide Check Test		X		42
DVH	0220	Divide or Halt		X	X	24
DVP	0221	Divide or Proceed		X	X	24
ECTM	—0760..0006	Enter Copy Trap Mode		X		66
EFTM	—0760..0002	Enter Floating Trap Mode		X		66
ENB	0564	Enable from Y		X	X	67
ENK	0760..0004	Enter Keys		X	X	35
ERA	0322	Exclusive OR to Accumulator		X	X	49
ESNT	—0021	Enter Storage Null. and Transfer		X	X	65
ESTM	—0760..0005	Enter Select Trap Mode		X		66
ETM	0760..0007	Enter Trapping Mode		X		36
ETT	—0760..xxxx	End of Tape Test		X		41
FAD	0300	Floating Add	3	X	X	26
FAM	0304	Floating Add Magnitude	3	X	X	27
FDH	0240	Floating Divide or Halt	5	X	X	30
FDP	0241	Floating Divide or Proceed	5	X	X	30
FMP	0260	Floating Multiply	1	X	X	29
FRN	0760..0011	Floating Round		X		28
FSB	0302	Floating Subtract	3	X	X	27
FSM	0306	Floating Subtract Magnitude	3	X	X	28
HPR	0420	Halt and Proceed				35
HTR	0000	Halt and Transfer		X	X	36
IIA	0041	Invert Indicators from AC				53
IIL	—0051	Invert Indicators of Left Half				53
IIR	0051	Invert Indicators of Right Half				53
IIS	0440	Invert Indicators from Storage		X	X	53
IOT	0760..0005	Input-Output Check Test		X	X	42
LAC	0535	Load Complement of Address in Index				45
LAS	—0340	Logical Compare Accumulator with Storage		X	X	43
LBT	0760..0001	Low- Order Bit Test		X		42
LCHA	0544	Load Channel A	8	X	X	61
LCHB	—0544	Load Channel B	8	X	X	61
LCHC	0545	Load Channel C	8	X	X	61
LCHD	—0545	Load Channel D	8	X	X	61
LCHE	0546	Load Channel E	8	X	X	61
LCHF	—0546	Load Channel F	8	X	X	61
LCHG	0547	Load Channel G	8	X	X	61
LCHH	—0547	Load Channel H	8	X	X	61
LDC	—0535	Load Complement of Decrement in XR				45
LDI	0441	Load Indicators		X	X	51
LDQ	0560	Load MQ		X	X	33
LFT	—0054	Left Half Indicators, Off Test				55
LFTM	—0760..0004	Leave Floating Trap Mode		X		66
LGL	—0763	Logical Left Shift	7	X		32
LGR	—0765	Logical Right Shift	7	X		32
LLS	0763	Long Left Shift	7	X		32
LNT	—0056	Left Half Indicators, On Test				54
LRS	0765	Long Right Shift	7	X		32
LSNM	—0760..0010	Leave Storage Nullification Mode		X		65
LTM	—0760..0007	Leave Trapping Mode		X		37
LXA	0534	Load Index from Address				45
LXD	—0534	Load Index from Decrement				45
MPR	—0200	Multiply and Round	1	X	X	22
MPY	0200	Multiply	1	X	X	22
MSE	—0760	Minus Sense		X		41
NOP	0761	No Operation				35
NZT	—0520	Storage Not-Zero Test		X	X	43
OAI	0043	OR Accumulator to Indicators				51
OFT	0444	Off Test for Indicators		X	X	54
ONT	0446	On Test for Indicators		X	X	54
ORA	—0501	OR to Accumulator		X	X	48
ORS	—0602	OR to Storage		X	X	48
OSI	0442	OR Storage to Indicators		X	X	51
PAC	0737	Place Complement of Address in XR				46
PAI	0044	Place Accumulator in Indicators				50
PAX	0734	Place Address in XR				46
PBT	—0760..0001	P-bit Test		X		42
PDC	—0737	Place Complement of Decrement in XR				46
PDX	—0734	Place Decrement in Index				46
PIA	—0046	Place Indicator in Accumulator				51
PSE	0760	Plus Sense		X		40
PXA	0754	Place Index in Address				47
PXD	—0754	Place Index in Decrement				47
RCHA	0540	Reset and Load Channel A		X	X	60
RCHB	—0540	Reset and Load Channel B		X	X	60
RCHC	0541	Reset and Load Channel C		X	X	60
RCHD	—0541	Reset and Load Channel D		X	X	60
RCHE	0542	Reset and Load Channel E		X	X	60
RCHF	—0542	Reset and Load Channel F		X	X	60
RCHG	0543	Reset and Load Channel G		X	X	60
RCHH	—0543	Reset and Load Channel H		X	X	60
RCT	0760..0014	Restore Channel Traps		X		65
RDCA	0760..1352	Reset Data Channel A		X		59
RDCB	0760..2352	Reset Data Channel B		X		59
RDCC	0760..3352	Reset Data Channel C		X		59
RDCD	0760..4352	Reset Data Channel D		X		59
RDCE	0760..5352	Reset Data Channel E		X		59
RDCF	0760..6352	Reset Data Channel F		X		59
RDCG	0760..7352	Reset Data Channel G		X		59
RDCH	0760.10352	Reset Data Channel H		X		59
RDS	0762	Read Select	8	X		58
REW	0772	Rewind	8	X		59
RFT	0054	Right Half Indicators, Off Test				55
RIA	—0042	Reset Indicators from Accumulator				52
RICA	0760..1350	Reset Channel A		X		68
RICB	0760..2350	Reset Channel B		X		68
RICC	0760..3350	Reset Channel C		X		68
RICD	0760..4350	Reset Channel D		X		68
RICE	0760..5350	Reset Channel E		X		68
RICF	0760..6350	Reset Channel F		X		68
RICG	0760..7350	Reset Channel G		X		68
RICH	0760.10350	Reset Channel H		X		68

Alphabetic *(Continued)*

OPERATION CODE		INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
ALPHA	OCTAL					
RIL	—0057	Reset Indicators of Left Half				52
RIR	0057	Reset Indicators of Right Half				52
RIS	0445	Reset Indicators from Storage		X	X	52
RND	0760..0010	Round		X		22
RNT	0056	Right Half Indicators, On Test				54
RQL	—0773	Rotate MQ Left	7	X		32
RSCA	+0540	Reset and Start Channel A		X	X	67
RSCB	—0540	Reset and Start Channel B		X	X	67
RSCC	+0541	Reset and Start Channel C		X	X	67
RSCD	—0541	Reset and Start Channel D		X	X	67
RSCE	+0542	Reset and Start Channel E		X	X	67
RSCF	—0542	Reset and Start Channel F		X	X	67
RSCG	+0543	Reset and Start Channel G		X	X	67
RSCH	—0543	Reset and Start Channel H		X	X	67
RUN	—0772	Rewind and Unload		X		59
SBM	—0400	Subtract Magnitude		X	X	21
SCHA	0640	Store Channel A		X	X	67
SCHB	—0640	Store Channel B		X	X	67
SCHC	0641	Store Channel C		X	X	67
SCHD	—0641	Store Channel D		X	X	67
SCHE	0642	Store Channel E		X	X	67
SCHF	—0642	Store Channel F		X	X	67
SCHG	0643	Store Channel G		X	X	67
SCHH	—0643	Store Channel H		X	X	67
SDN	0776	Set Density		X		59
SIL	—0055	Set Indicator of Left Half				51
SIR	0055	Set Indicator of Right Half				52
SLQ	—0620	Store Left Half MQ		X	X	33
SLW	0602	Store Logical Word		X	X	33
SSM	—0760..0003	Set Sign Minus		X		50
SSP	0760..0003	Set Sign Plus		X		50
STA	0621	Store Address		X	X	34
STCA	+0544	Start Channel A		X	X	67
STCB	—0544	Start Channel B		X	X	67
STCC	+0545	Start Channel C		X	X	67
STCD	—0545	Start Channel D		X	X	67
STCE	+0546	Start Channel E		X	X	67
STCF	—0546	Start Channel F		X	X	67
STCG	+0547	Start Channel G		X	X	67
STCH	—0547	Start Channel H		X	X	67
STD	0622	Store Decrement		X	X	34
STI	0604	Store Indicators		X	X	51
STL	—0625	Store Instruction Location Counter		X	X	34
STO	0601	Store		X	X	33
STP	0630	Store Prefix		X	X	33
STQ	—0600	Store MQ		X	X	33
STR	—1000	Store Location and Trap				34
STT	0625	Store Tag		X	X	34
STZ	0600	Store Zero		X	X	34
SUB	0402	Subtract		X	X	21
SXA	0634	Store Index in Address				46
SXD	—0634	Store Index in Decrement				46
TCNA	—0060	Transfer on DSC A Not in Operation		X	X	44
TCNB	—0061	Transfer on DSC B Not in Operation		X	X	44
TCNC	—0062	Transfer on DSC C Not in Operation		X	X	44
TCND	—0063	Transfer on DSC D Not in Operation		X	X	44
TCNE	—0064	Transfer on DSC E Not in Operation		X	X	44
TCNF	—0065	Transfer on DSC F Not in Operation		X	X	44
TCNG	—0066*	Transfer on DSC G Not in Operation		X	X	44
TCNH	—0067*	Transfer on DSC H Not in Operation		X	X	44
TCOA	0060	Transfer on DSC A in Operation		X	X	43

Alphabetic *(Continued)*

OPERATION CODE		INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
ALPHA	OCTAL					
TCOB	0061	Transfer on DSC B in Operation		X	X	43
TCOC	0062	Transfer on DSC C in Operation		X	X	43
TCOD	0063	Transfer on DSC D in Operation		X	X	43
TCOE	0064	Transfer on DSC E in Operation		X	X	43
TCOF	0065	Transfer on DSC F in Operation		X	X	43
TCOG	0066	Transfer on DSC G in Operation		X	X	43
TCOH	0067	Transfer on DSC H in Operation		X	X	43
TEFA	0030	Transfer on DSC A End of File		X	X	44
TEFB	—0030	Transfer on DSC B End of File		X	X	44
TEFC	0031	Transfer on DSC C End of File		X	X	44
TEFD	—0031	Transfer on DSC D End of File		X	X	44
TEFE	0032	Transfer on DSC E End of File		X	X	44
TEFF	—0032	Transfer on DSC F End of File		X	X	44
TEFG	0033	Transfer on DSC G End of File		X	X	44
TEFH	—0033	Transfer on DSC H End of File		X	X	44
TIF	0046	Transfer if Indicators Off		X	X	53
TIO	0042	Transfer if Indicators On		X	X	53
TIX	2000	Transfer on Index				40
TLQ	0040	Transfer on Low MQ		X	X	39
TMI	—0120	Transfer on Minus		X	X	38
TNO	—0140	Transfer on No Overflow		X	X	38
TNX	—2000	Transfer on No Index				40
TNZ	—0100	Transfer on No Zero		X	X	37
TOV	0140	Transfer on Overflow		X	X	38
TPL	0120	Transfer on Plus		X	X	38
TQO	0161	Transfer on Quotient Overflow		X	X	38
TQP	0162	Transfer on MQ Plus		X	X	38
TRA	0020	Transfer		X	X	36
TRCA	0022	Transfer on DSC A Redundancy Check		X	X	44
TRCB	—0022	Transfer on DSC B Redundancy Check		X	X	44
TRCC	0024	Transfer on DSC C Redundancy Check		X	X	44
TRCD	—0224	Transfer on DSC D Redundancy Check		X	X	44
TRCE	0026	Transfer on DSC E Redundancy Check		X	X	44
TRCF	—0026	Transfer on DSC F Redundancy Check		X	X	44
TRCG	0027	Transfer on DSC G Redundancy Check		X	X	44
TRCH	—0027	Transfer on DSC H Redundancy Check		X	X	44
TSX	0074	Transfer and Set Index				39
TTR	0021	Trap Transfer		X	X	37
TXH	3000	Transfer on Index High				39
TXI	1000	Transfer with XR Incremented				39
TXL	—3000	Transfer on XR Low or Equal				40
TZE	0100	Transfer on Zero		X	X	37
UAM	—0304	Unnormalized Add Magnitude	4	X	X	28
UFA	—0300	Unnormalized Floating Add	4	X	X	27
UFM	—0260	Unnormalized Floating Multiply	1	X	X	29

Alphabetic *(Continued)*

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
UFS	—0302	Unnormalized Floating Subtract	4	X	X	28
USM	—0306	Unnormalized Subtract Magnitude	4	X	X	28
VDH	0224	Variable Length Divide or Halt	2	X		24
VDP	0225	Variable Length Divide or Proceed	2	X		24
VLM	0204	Variable Length Multiply	1,2	X		22
WEF	0770	Write End of File	8	X		59
WRS	0766	Write Select	8	X		58
XCA	0131	Exchange AC and MQ				34
XCL	—0130	Exchange Logical AC and MQ				34
XEC	0522	Execute		X	X	36
ZET	0520	Storage Zero Test		X	X	43

Numerical Listing

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
HTR	0000	Halt and Transfer		X	X	36
TRA	0020	Transfer		X	X	36
TTR	0021	Trap Transfer		X	X	37
ESNT	—0021	Enter Storage Null. and Transfer		X	X	65
TRCA	0022	Transfer on DSC A Redundancy Check		X	X	44
TRCB	—0022	Transfer on DSC B Redundancy Check		X	X	44
TRCC	0024	Transfer on DSC C Redundancy Check		X	X	44
TRCD	—0024	Transfer on DSC D Redundancy Check		X	X	44
TRCE	0026	Transfer on DSC E Redundancy Check		X	X	44
TRCF	—0026	Transfer on DSC F Redundancy Check		X	X	44
TRCG	0027	Transfer on DSC G Redundancy Check		X	X	44
TRCH	—0027	Transfer on DSC H Redundancy Check		X	X	44
TEFA	0030	Transfer on DSC A End of File		X	X	44
TEFB	—0030	Transfer on DSC B End of File		X	X	44
TEFC	0031	Transfer on DSC C End of File		X	X	44
TEFD	—0031	Transfer on DSC D End of File		X	X	44
TEFE	0032	Transfer on DSC E End of File		X	X	44
TEFF	—0032	Transfer on DSC F End of File		X	X	44
TEFG	0033	Transfer on DSC G End of File		X	X	44
TEFH	—0033	Transfer on DSC H End of File		X	X	44
TLQ	0040	Transfer on Low MQ		X	X	39
IIA	0041	Invert Indicators from AC				53
TIO	0042	Transfer if Indicators On		X	X	53
RIA	—0042	Reset Indicators from AC				52
OAI	0043	OR Accumulator to Indicators				51
PAI	0044	Place Accumulator in Indicators				50
TIF	0046	Transfer if Indicators Off		X	X	53
PIA	—0046	Place Indicators in Accumulator				51
IIR	0051	Invert Indicators of Right Half				53
IIL	—0051	Invert Indicators of Left Half				53
RFT	0054	Right Half Indicators, Off Test				55
LFT	—0054	Left Half Indicators, Off Test				55
SIR	0055	Set Indicator of Right Half				52

Numerical *(Continued)*

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
SIL	—0055	Set Indicator of Left Half				51
RNT	0056	Right Half Indicators, On Test				54
LNT	—0056	Left Half Indicators, On Test				54
RIR	0057	Reset Indicators of Right Half				52
RIL	—0057	Reset Indicators on Left Half				52
TCNA	—0060	Transfer DSC A Not in Operation		X	X	44
TCNB	—0061	Transfer DSC B Not in Operation		X	X	44
TCNC	—0062	Transfer DSC C Not in Operation		X	X	44
TCND	—0063	Transfer DSC D Not in Operation		X	X	44
TCNE	—0064	Transfer DSC E Not in Operation		X	X	44
TCNF	—0065	Transfer DSC F Not in Operation		X	X	44
TCNG	—0066	Transfer DSC G Not in Operation		X	X	44
TCNH	—0067	Transfer DSC H Not in Operation		X	X	44
TCOA	0060	Transfer DSC A in Operation		X	X	43
TCOB	0061	Transfer DSC B in Operation		X	X	43
TCOC	0062	Transfer DSC C in Operation		X	X	43
TCOD	0063	Transfer DSC D in Operation		X	X	43
TCOE	0064	Transfer DSC E in Operation		X	X	43
TCOF	0065	Transfer DSC F in Operation		X	X	43
TCOG	0066	Transfer DSC G in Operation		X	X	43
TCOH	0067	Transfer DSC H in Operation		X	X	43
TSX	0074	Transfer and Set Index				39
TZE	0100	Transfer on Zero		X	X	37
TNZ	—0100	Transfer on No Zero		X	X	37
CVR	0114	Convert by Replacement from AC	6			56
CAQ	—0114	Convert by Addition from MQ	6			57
TPL	0120	Transfer on Plus		X	X	38
TMI	—0120	Transfer on Minus		X	X	38
XCL	—0130	Exchange Logical Accumulator and MQ				34
XCA	0131	Exchange Accumulator and MQ				34
TOV	0140	Transfer on Overflow		X	X	38
TNO	—0140	Transfer on No Overflow		X	X	38
CRQ	—0154	Convert by Replacement from MQ	6			56
TQO	0161	Transfer on Quotient Overflow		X	X	38
TQP	0162	Transfer on MQ Plus		X	X	38
MPY	0200	Multiply	1	X	X	22
MPR	—0200	Multiply and Round	1	X	X	22
VLM	0204	Variable Length Multiply	1,2	X		22
DVH	0220	Divide or Halt		X	X	24
DVP	0221	Divide or Proceed		X	X	24
VDH	0224	Variable Length Divide or Halt	2	X		24
VDP	0225	Variable Length Divide or Proceed	2	X		24
FDH	0240	Floating Divide or Halt	5	X	X	30
FDP	0241	Floating Divide or Proceed	5	X	X	30
FMP	0260	Floating Multiply	1	X	X	29
UFM	—0260	Unnormalized Floating Multiply	1	X	X	28

Numerical *(Continued)*

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
FAD	0300	Floating Add	3	X	X	26
UFA	—0300	Unnormalized Floating Add	4	X	X	27
FSB	0302	Floating Subtract	3	X	X	27
UFS	—0302	Unnormalized Floating Subtract	4	X	X	28
FAM	0304	Floating Add Magnitude	3	X	X	27
UAM	—0304	Unnormalized Add Magnitude	4	X	X	28
FSM	0306	Floating Subtract Magnitude	3	X	X	28
USM	—0306	Unnormalized Subtract Magnitude	4	X	X	28
ANS	0320	AND to Storage		X	X	48
ANA	—0320	AND to Accumulator		X	X	48
ERA	0322	Exclusive OR to Accumulator		X	X	49
CAS	0340	Compare Accumulator with Storage		X	X	43
LAS	—0340	Logical Compare AC with Storage		X	X	43
ACL	0361	Add and Carry Logical Word		X	X	21
ADD	0400	Add		X	X	20
SBM	—0400	Subtract Magnitude		X	X	21
ADM	0401	Add Magnitude		X	X	20
SUB	0402	Subtract		X	X	21
HPR	0420	Halt and Proceed				35
IIS	0440	Invert Indicators from Storage		X	X	53
LDI	0441	Load Indicators		X	X	51
OSI	0442	OR Storage to Indicators		X	X	51
OFT	0444	Off Test for Indicators		X	X	54
RIS	0445	Reset Indicators from Storage		X	X	52
ONT	0446	On Test for Indicators		X	X	54
CLA	0500	Clear and Add		X	X	20
CAL	—0500	Clear and Add Logical Word		X	X	20
ORA	—0501	OR to Accumulator		X	X	48
CLS	0502	Clear and Subtract		X	X	20
ZET	0520	Storage Zero Test		X	X	43
NZT	—0520	Storage Not-Zero Test		X	X	43
XEC	0522	Execute		X	X	36
LXA	0534	Load Index from Address				45
LXD	—0534	Load Index from Decrement				45
LAC	0535	Load Complement of Address in XR				45
LDC	—0535	Load Complement of Decrement in XR				45
RCHA	0540	Reset and Load Channel A		X	X	60
RCHB	—0540	Reset and Load Channel B		X	X	60
RCHC	0541	Reset and Load Channel C		X	X	60
RCHD	—0541	Reset and Load Channel D		X	X	60
RCHE	0542	Reset and Load Channel E		X	X	60
RCHF	—0542	Reset and Load Channel F		X	X	60
RCHG	0543	Reset and Load Channel G		X	X	60
RCHH	—0543	Reset and Load Channel H		X	X	60
RSCA	+0540	Reset and Start Channel A		X	X	67
RSCB	—0540	Reset and Start Channel B		X	X	67
RSCC	+0541	Reset and Start Channel C		X	X	67
RSCD	—0541	Reset and Start Channel D		X	X	67
RSCE	+0542	Reset and Start Channel E		X	X	67
RSCF	—0542	Reset and Start Channel F		X	X	67
RSCG	+0543	Reset and Start Channel G		X	X	67
RSCH	—0543	Reset and Start Channel H		X	X	67
LCHA	0544	Load Channel A	8	X	X	61
LCHB	—0544	Load Channel B	8	X	X	61
LCHC	0545	Load Channel C	8	X	X	61
LCHD	—0545	Load Channel D	8	X	X	61
LCHE	0546	Load Channel E	8	X	X	61
LCHF	—0546	Load Channel F	8	X	X	61
LCHG	0547	Load Channel G	8	X	X	61
LCHH	—0547	Load Channel H	8	X	X	61
STCA	+0544	Start Channel A		X	X	67
STCB	—0544	Start Channel B		X	X	67
STCC	+0545	Start Channel C		X	X	67
STCD	—0545	Start Channel D		X	X	67
STCE	+0546	Start Channel E		X	X	67

Numerical *(Continued)*

OPERATION CODE ALPHA	OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
STCF	—0546	Start Channel F		X	X	67
STCG	+0547	Start Channel G		X	X	67
STCH	—0547	Start Channel H		X	X	67
LDQ	0560	Load MQ		X	X	33
ENB	0564	Enable		X	X	65
STZ	0600	Store Zero		X	X	34
STQ	—0600	Store MQ		X	X	33
STO	0601	Store		X	X	33
SLW	0602	Store Logical Word		X	X	33
ORS	—0602	OR to Storage		X	X	48
STI	0604	Store Indicators		X	X	51
SLQ	—0620	Store Left Half MQ		X	X	33
STA	0621	Store Address		X	X	34
STD	0622	Store Decrement		X	X	34
STT	0625	Store Tag		X	X	34
STL	—0625	Store Instruction Location Counter		X	X	34
STP	0630	Store Prefix		X	X	33
SXA	0634	Store Index in Address				46
SXD	—0634	Store Index in Decrement				46
SCHA	0640	Store Channel A		X	X	60
SCHB	—0640	Store Channel B		X	X	60
SCHC	0641	Store Channel C		X	X	60
SCHD	—0641	Store Channel D		X	X	60
SCHE	0642	Store Channel E		X	X	60
SCHF	—0642	Store Channel F		X	X	60
SCHG	0643	Store Channel G		X	X	60
SCHH	—0643	Store Channel H		X	X	60
PAX	0734	Place Address in Index				46
PDX	—0734	Place Decrement in Index				46
PAC	0737	Place Complement of Address in XR				46
PDC	—0737	Place Complement of Decrement in XR				46
PXA	0754	Place Index in Address				47
PXD	—0754	Place Index in Decrement				47
PSE	0760	Plus Sense		X		40
MSE	—0760	Minus Sense		X		41
CLM	0760..0000	Clear Magnitude		X		49
LBT	0760..0001	Low-Order Bit Test		X		42
PBT	—0760..0001	P-bit Test		X		42
CHS	0760..0002	Change Sign		X		49
EFTM	—0760..0002	Enter Floating Trap Mode		X		66
SSP	0760..0003	Set Sign Plus		X		50
SSM	—0760..0003	Set Sign Minus		X		50
ENK	0760..0004	Enter Keys		X		35
LFTM	—0760..0004	Leave Floating Trap Mode		X		66
IOT	0760..0005	Input-Output Check Test		X		42
ESTM	—0760..0005	Enter Select Trap Mode		X		66
COM	0760..0006	Complement Magnitude		X		49
ECTM	—0760..0006	Enter Copy Trap Mode		X		66
ETM	0760..0007	Enter Trapping Mode		X		36
LTM	—7060..0007	Leave Trapping Mode		X		37
RND	0760..0010	Round		X		22
LSNM	—0760..0010	Leave Storage Nullification Mode		X		65
FRN	0760..0011	Floating Round		X		28
DCT	0760..0012	Divide Check Test		X		42
RCT	0760..0014	Restore Channel Traps		X		65
RDCA	0760..1352	Reset Data Channel A		X		59
RDCB	0760..2352	Reset Data Channel B		X		59
RDCC	0760..3352	Reset Data Channel C		X		59
RDCD	0760..4352	Reset Data Channel D		X		59
RDCE	0760..5352	Reset Data Channel E		X		59
RDCF	0760..6352	Reset Data Channel F		X		59
RDCG	0760..7352	Reset Data Channel G		X		59
RDCH	0760.10352	Reset Data Channel H		X		59
RICA	0760..1350	Reset Channel A		X		68
RICB	0760..2350	Reset Channel B		X		68
RICC	0760..3350	Reset Channel C		X		68
RICD	0760..4350	Reset Channel D		X		68
RICE	0760..5350	Reset Channel E		X		68
RICF	0760..6350	Reset Channel F		X		68

Numerical *(Continued)*

OPERATION CODE ALPHA	OPERATION CODE OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
RICG	0760..7350	Reset Channel G		X		68
RICH	0760.10350	Reset Channel H		X		68
NOP	0761	No Operation				35
RDS	0762	Read Select	8	X		58
LLS	0763	Long Left Shift	7	X		32
LGL	—0763	Logical Left Shift	7	X		32
BSR	0764	Backspace Record	8	X		58
BSF	—0764	Backspace File	8	X		58
LRS	0765	Long Right Shift	7	X		32
LGR	—0765	Logical Right Shift	7	X		32
WRS	0766	Write Select	8	X		58
ALS	0767	Accumulator Left Shift	7	X		31
WEF	0770	Write End of File	8	X		59
ARS	0771	Accumulator Right Shift	7	X		32
REW	0772	Rewind	8	X		59

Numerical *(Continued)*

OPERATION CODE ALPHA	OPERATION CODE OCTAL	INSTRUCTION	MODIF'N	INDEXABLE	INDIRECTLY ADDRESSABLE	PAGE
RUN	—0772	Rewind and Unload		X		59
RQL	—0773	Rotate MQ Left	7	X		32
AXT	0774	Address to Index True				45
AXC	—0774	Address to Index Complemented				46
SDN	0776	Set Density		X		59
TXI	1000	Transfer with XR Incremented				39
STR	—1000	Store Location and Trap				34
TIX	2000	Transfer on Index				40
TNX	—2000	Transfer on No Index				40
TXH	3000	Transfer on Index High				39
TXL	—3000	Transfer on XR Low or Equal				40

Appendix G. Instructions by Operation Group

Instructions by Operation Group *(Cont'd)*

Index

A22-6528-4

IBM 7090 Printed in U.S.A. A22-6528-4

IBM

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