Copy 25

THE ERA 1103

COMPUTER SYSTEM

(TYPE 2300C1)

VOLUME IV

SECTION 6. PROGRAMMING

PX 71871

Remington Rand
INC
ENGINEERING RESEARCH ASSOCIATES DIVISION
1902 WEST MINNEHAHA AVE. ST. PAUL W4, MINNESOTA

TABLE OF CONTENTS
VOLUME 4

SECTION 6
PROGRAMMING

TABLE OF CONTENTS (continued)
VOLUME 4

PX 71671

TABLE OF CONTENTS (concluded)
VOLUME 4

PX 71871

SECTION 6

PROGRAMMING

1. INTRODUCTION.

A general purpose digital computer is capable of executing the basic arithmetic operations, i.e., addition, subtraction, multiplication, and division, plus performing internal data handling operations, logical operations, and input-output operations. Any problem that can be solved by numerical techniques can be handled and solved by computer operations. The given problem must be analyzed and resolved into a collection of smaller problems, each of which can be solved by the application of the basic arithmetic operations. The necessary operations are performed by a computer as the resultant effect of its control section receiving and interpreting the coded instructions with such operations as their functions. Portions of the coded instructions represent operands, the numbers which are involved in and necessary for the execution of the instructions. A list of such computer instructions with the appropriate operands is called a program. During the execution of a program, instructions are removed from computer storage, one at a time, in the order in which they are required, and placed in computer control which directs the execution of each instruction.

Information upon which the computer operates is numerically coded according to binary number notation. Thus an instruction or an operand is represented by a series combination of "0's" and "1's". The quantity of binary digits ("bits") permissible is determined by the physical characteristics of the computer: the information to be represented is restricted to a certain number of bits according to the medium of representation. A binary digit is represented by the state of a bi-stable element, the quantity of these elements thus determining the size of a binary coded "word", i.e., an operand or 36-bit instruction. The storage systems for the 1103 provide internal storage facilities for 36-bit words (including information coded in less than 36 significant bits). Information is also represented internally in "registers" consisting of a pre-designated number of stages of bi-stable elements. A "36-bit register" is capable of representing 36 bits of coded information; a "72-bit register" is capable of representing 72 bits.

The 1103 has in its repertoire 44 instructions each of which is individually coded as 36 bits, $i_{35} \ldots i_0$. The left-most six bits of an instruction, $i_{35} \ldots i_{30}$, represent its operation code, the portion of the instruction which "describes" to the computer control which particular operation is to be executed. The remaining 30 bits of the instruction are grouped as $i_{29} \ldots i_{15}$ and $i_{14} \ldots i_0$ and are designated as the u-address portion and the v-address portion of the instruction, respectively. These are the portions of the instruction which represent the operands (by referencing their location in storage, for the most part) with which the operation is concerned. An explanation of what the execution of each instruction accomplishes, with the location being given of any operands used during this execution and any pertinent results derived from this execution, is designated as the function of the instruction.

2. THE COMPUTER.

a. GENERAL DESCRIPTION. - The representation of numbers in the 1103 and the operations performed on them cannot be discussed without some previous knowledge of the logic of the 1103. The overall logic of a computing system is a synthesis of the methods, inherent to a particular computer design, which enable the performance of prescribed operations. This text will form an outline of the logic of the 1103 by explaining briefly the different components of the computer and their functions.

The components of the computer may be grouped, in general, into five basic sections according to their function in the overall operation of the computer. These groupings with their functions are as follows:

(1) INPUT AND OUTPUT components which provide a means of communication between the computer and the operator. Input components receive the coded information from external equipment and effect its insertion into storage. Output components deliver the results of computer operations to external equipment.

An all-inclusive listing of the components of the systems which enable input and/or output operations would include external equipment, external equipment control, and certain of the computer registers with their control circuitry.

(2) STORAGE devices in which information is set aside for immediate or future use.

(3) ARITHMETIC section in whose registers arithmetic and logical operations are performed and operands and results are temporarily stored.

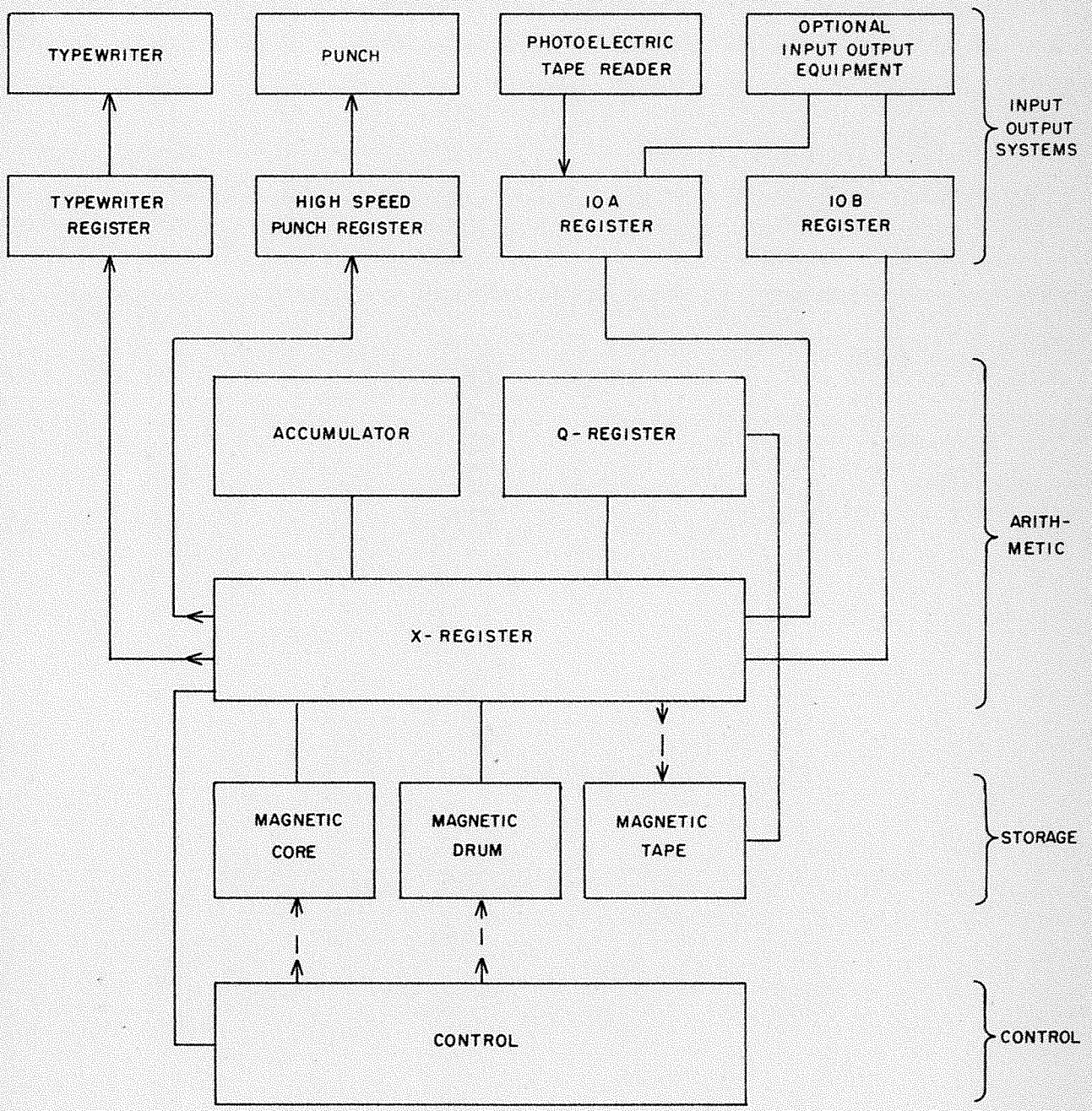
(4) CONTROL components which direct the operations of the computer.

(5) POWER system which provides regulated voltages to all other sections of the computer.

Transmission of information between components of the first four sections above is internally routed, in most cases, through the X-Register, a 36-bit register capable of temporary storage of information. The X-Register also functions as a component of the arithmetic section. The storage and arithmetic sections have in common two other registers, the Accumulator, a 72-bit register, and the Q-Register, a 36-bit register. These may function as temporary storage devices or registers in which arithmetic operations are performed by digital manipulation. In addition the Q-Register is used in some cases as an assembly register for information being placed in it in segments of less than 36-bit word size.

A simplified block diagram showing the interrelationship of the components of the computer is presented in Figure 6-1.

b. BASIC COMPONENTS. - Certain of the aforementioned sections, namely the



Solid lines connecting blocks indicate the routing of binary information. Dashed lines carry binary information used for storage reference purposes. Information may be routed in both directions unless arrows on a line indicate a one-way transfer.

Lines which carry control signals between the above blocks are not shown on this diagram.

Figure 6-1. Simplified Block Diagram of the 1103 Computer System

storage, arithmetic, and control sections, and the manner in which the components of each function during computer operation, are discussed in the following sub-paragraphs. (The Input and Output Systems are discussed in Paragraph 4.)

(1) STORAGE DEVICES. - The information which is held in storage consists of instructions which are to be executed and the operands needed by these instructions for their execution. The location of each instruction or operand is referred to as its "address".

There are four classes of storage locations which are individually addressed: Magnetic Drum storage, MD: Magnetic Core storage, MC; the Accumulator, A; and the Q-Register, Q. These classes have the following addresses assigned to them:

Storage Class	Octal Equivalents of Addresses	Storage Space: Number of Words
MC	00000-01777	1024
Q	10000-17777	1
A	20000-27777	1 double length
MD (Group 4)	40000-47777	4096
MD (Group 5)	50000-57777	4096
MD (Group 6)	60000-67777	4096
MD (Group 7)	70000-77777	4096

Additional storage is provided by the Magnetic Tape Storage System, MT, in which the information is not individually addressed.

(a) ADDRESSED STORAGE LOCATIONS. - Information is acquired from storage in MC, MD, A, or Q by first placing the address of the instruction or operand desired in a section of computer control where the storage class is determined. Following this, the address is transmitted to the locating control of the proper storage class where the specific storage location is found (if the storage class is MC or MD), after which the information at this location is transmitted to the X-Register. Operations of this kind are referred to as "reading" operations. "Writing" operations, or the transfer of information to a storage location, are accomplished in a similar manner with the information in the X-Register being placed at a storage address as located by the control circuitry.

1 MAGNETIC DRUM STORAGE SYSTEM (MD). - The Magnetic Drum Storage System provides medium-access binary storage. Digital information is stored in the form of magnetized areas on the surface of a continuously rotating cylinder called a magnetic drum. The medium of storage is a magnetized dipole having either of two polarity orientations in the lateral (or peripheral) direction. For all practical purposes, information recorded on the drum is stored permanently. It may, however, be removed by special erase techniques or it may be replaced by simply writing new information over it. Reading from the drum does not alter the contents of the location read in any way.

Each individual storage location is identified by specifying its angular and axial coordinates on the drum surface. A word is stored in 36 axially adjacent positions on the drum with 4096 such angular word locations available.

In use are four groups as noted previously, thus giving the Magnetic Drum Storage System a capacity of 16,384 36-bit computer words. When a word or a portion of a word is to be transmitted to or from the magnetic drum, all the bits to be transmitted are handled simultaneously, i.e., in parallel. Information may be recorded or read in any given area only once during each drum revolution, resulting in a maximum access time of 34 milliseconds.

2. MAGNETIC CORE STORAGE SYSTEM (MC). - The Magnetic Core Storage System provides rapid-access storage for a total of 1024 36-bit words. Each core is a bistable element capable of storing a "1" or a "0", dependent upon the direction of magnetization of the core. The cores are arranged in a 32 x 32 matrix with 36 such matrices. The 36 digits of a given word are represented by the state of corresponding cores, one in each of the 36 matrices. Reading and writing operations of a word, or portion of a word, are performed in a parallel mode with a simultaneous transmission of bits. Certain sequences of pulses on wires through the cores, producing magnetizing forces of a certain polarity, are used to perform the reading and writing operations. Reading from MC does not alter the contents of the location read in any way.

Magnetic Core storage is non-volatile, comparable to non-volatile storage in the Magnetic Drum Storage System.

3. A AND Q AS STORAGE MEDIA. - The Accumulator and Q-Register are available as temporary storage registers since they may be referenced by addresses. The Q-Register is normally addressed as octal 10000 although any of the addresses 10000-17777 are permissible. Similarly, the Accumulator is normally addressed as 20000 with the addresses 20000-27777 being permissible.

(b) MAGNETIC TAPE STORAGE SYSTEM (MT). - The storage media of the Magnetic Tape Storage System are four reels of magnetic tape on which information is recorded in a manner similar to that described for the magnetic drum. Words stored on magnetic tape are grouped in blocks, each block consisting of 32 words and each reel of tape capable of storing 2048 blocks. Thus the MT Storage System may store as many as 262,144 words on its four reels of tape. However, individual words in MT storage are not individually addressed, but located for reading and writing operations by properly coded programmed instructions. The coded portion of the instruction designating the reel of tape and the blocks in this reel from or to which information is to be transmitted is placed in the X-Register. From the X-Register it is transmitted to the locating control circuitry of the MT Storage System. This control positions the designated reel of tape for reading or writing operations. A reading operation transmits information from the magnetic tape to magnetic core storage, via the Q-Register and X-Register; a writing operation places information from magnetic core storage on the magnetic tape, the information being transmitted via the X-Register and Q-Register.

(2) ARITHMETIC SECTION. - The components of the arithmetic section of the computer are used in performing numerical operations by digital manipulations. The operations of addition, subtraction, multiplication, and division are effected within the Accumulator and the Q-Register, with the X-Register available to provide temporary storage for operands. These operations are made possible and restricted to the modulus of the respective

PX 71871

register involved by the "end-around borrow" property of the Accumulator and the "circular shifting" property of the Accumulator and the Q-Register, both of which have parallel stages of bi-stable elements.

The circular shifting property of a register allows a number derived by a simultaneous left shift of each of the bits representing the number to retain a modular value of the system that can be represented by the particular register. Thus a number derived by a left shift of its expression in bits, $b_n \ldots b_i \ldots b_o$, as represented by the stages $S_n \ldots S_i \ldots S_o$, has the modular value of its expression in bits $b_{n-1} \ldots b_{i-1} \ldots b_n$, as represented by the same stages $S_n \ldots S_i \ldots S_o$. The value of i in connection with the Q-Register is $36 > i \geq 0$; as restricted by the Accumulator, i has the range $72 > i \geq 0$. The end-around borrow property restricts a remainder, while enabling the subtraction, to a modular value of the one's complement number system that can be represented by the Accumulator (modulus $2^{72}-1$). The subtractive process using the Accumulator is the basis of addition, subtraction, multiplication, and division as performed by the computer.

The placement of the operands and results of arithmetic operations during their execution is as follows: For the corresponding arithmetic operations, the X-Register holds the addend, subtrahend, multiplicand, and divisor; the Q-Register holds the multiplier, quotient, and logical multiplier; the Accumulator holds the sum, difference, product, and remainder.

(3) CONTROL COMPONENTS. - Each of the function groupings of the computer, input and output, storage, and arithmetic, has individual control systems which direct the operations of the section under their influence. These control systems are in turn directed in their operations by the main computer control. This overall influence exerted by computer control is necessary for time-wise reasons: an established sequence of internal actions is essential for the processing of any coded information. The computer control initiates and superintends these patterns of actions during their performance.

The main control section receives the instructions which the computer is to carry out, interprets them, and directs their execution with the operands specified. The computer must be manually started, but can stop automatically or be manually stopped. (In addition to being automatically controlled by a program of instructions, it can be manually controlled from the Supervisory Control Panel which contains all the necessary controls and indicators for manually operating the equipment.) The principal components of the control section are as follows:

(a) PROGRAM ADDRESS COUNTER. - The Program Address Counter, PAK, is a 15-stage additive counter. During computation PAK generates the consecutive addresses of the programmed instructions to be executed. The address in PAK is referred to each time an instruction word is to be obtained from the computer memory. The starting address of a computation may be manually inserted into PAK before the START (operation) button is pushed: if this is done, computation will begin by picking up the instruction stored at that address. If PAK is not manually pre-set, it will automatically be set to either MD address 40000 or MC address 00000 depending on other starting selections that are made. The operations which terminate the execution of each instruction extract from storage the next instruction to be executed (the address of which is held at that

time in PAK), after which the content of PAK is advanced by one. Thus during the termination of the instruction at address y, the instruction at y + 1 (the address held in PAK), is extracted from storage and PAK advanced to y + 2. If the instruction at address y indicated that the normal sequence of instructions be interrupted by a jump to an instruction not stored at a consecutive address, this instruction's address is inserted into PAK previous to the termination operations.

The generation of consecutive binary numbers in PAK is controlled by the following conditions in its physical structure: There is no communication (no possibility of a borrow), between the stages PAK_{10} and PAK_9 unless the stage PAK_{14} contains a value of one; thus PAK_{10} will not be affected by the advance of PAK after the contents of PAK_9 ... PAK_0 reach the value of $2^{10}-1$. The next advance of PAK after such a value is reached results in the contents of the stages PAK_9 ... PAK_0 being changed to zeroes. If PAK_{14} does contain a one, the contents of PAK may be increased until the contents of PAK_{13} ... PAK_0 reach the value of $2^{14}-1$. Then, since there is no communication between the stages PAK_{14} and PAK_{13}, the next advance of PAK will result in the contents of PAK_{13} ... PAK_0 being changed to zeroes with the value of one being left undisturbed in PAK_{14}. These "closed loop" systems effect the generation of successive addresses in PAK as follows: if an MC, Q, or A address is contained in PAK, the addresses can be advanced to (octal) 01777, 11777, or 21777, respectively, with the next advance of PAK resulting in its contents becoming (octal) 00000, 10000, or 20000, respectively. (The generation of consecutive A addresses in PAK is very improbable; if an attempt is made to take the next instruction to be executed from A, a fault occurs.) If any of the Magnetic Drum addresses, irregardless of the group, are represented in PAK, the addresses can be generated consecutively to 77777 with the next advance of PAK resulting in its contents becoming 40000.

(b) PROGRAM CONTROL REGISTERS. - The Program Control Registers, PCR, receive each instruction and temporarily store it during its execution. The registers consist of the Main Control Register, MCR, the U-Address Counter, UAK, and the V-Address Counter, VAK. Each instruction sent to PCR consists of a 6-bit operation code, which is stored in MCR; a 15-bit u-address portion, which is stored in UAK; and a 15-bit v-address portion, which is stored in VAK. Each instruction is obtained from some 36-bit storage location as specified by the Program Address Counter, PAK.

(c) MASTER CLOCK. - All the activities which take place within the computer, except for certain ones in the magnetic tape and output sections, are synchronized by a central timing system, called the Master Clock. During NORMAL computer operation, the clock generates 500 kc clock pulses based on timing pulses from the Magnetic Drum Storage System, and, after exerting certain controlling influences over them, supplies them to circuits throughout the computer. During TEST operations, a 500 kc oscillator may be used instead of the drum as the basic source of timing pulses. Circuits of the master clock system control the rate at which pulses leave the clock and also control their flow to the Main Pulse Distributor.

(d) MAIN PULSE DISTRIBUTOR. - The Main Pulse Distributor, MPD, receives clock pulses and distributes them in successive cycles of from four to eight pulses to the Command Timing Circuits. The distributor supplies each of the pulses cyclicly on its eight output lines. In an eight pulse cycle, all of the output lines are used and the pulses are designated, in the order of

PX 71871

their generation, MP0 through MP7. The selection of a particular cycle is made on the basis of the operation code held in the Main Control Register, MCR. Each code selects the cycle which will permit its performance in the least possible time.

(e) MAIN CONTROL TRANSLATOR - The principal translator of the Main Control Translator, MCT, receives a 6-bit operation code from MCR and produces accordingly a single prime operation code enable (indication to the computer that a certain instruction is to be executed). In CTC, enables from MCT are used in the selection of the sequence of commands which are needed to execute the instruction currently in MCR. In MPD, the MCT enables are used in the selection of the main pulse cycle required for the operation.

(f) COMMAND TIMING CIRCUITS. - The Command Timing Circuits, CTC, produce a discrete sequence of commands which execute the specified operation. The commands initiated are chosen (according to the operation code in MCR), by combining the operation enable from MCT and the pulse cycle received from MPD which consists of two or more of the pulses MP0 through MP5 and MP6 and MP7. Thus, it initiates the commands which execute the operation on pulses MP0 through MP5; reads the succeeding instruction from storage into the X-Register on MP6; then transfers the instruction from X into PCR on MP7.

c. NUMBER OPERATIONS.

(1) REPRESENTATION OF NUMBERS. - The modulus of a binary number system to be represented by k number of stages of bi-stable elements is 2^k. A negative number N of this system, modulus 2^k, is represented according to the one's complement system by $2^k-1-|N|$. (See Appendix 6A for further discussion of the representation of negative numbers.) Thus in MC and MD storage and the X and Q registers a negative binary number is expressed machine-wise as $2^{36}-1-|N|$, and in the Accumulator, as $2^{72}-1-|N|$. With the left-most stage used to indicate the sign of the number, the values possible to representation by such registers lie in the range with the limits of, respectively,

$$\pm(2^{35}-1), \text{ inclusive}$$

and

$$\pm(2^{71}-1), \text{ inclusive}$$

If the absolute value of a number is equal to or less than $2^{34}-1$ (or $2^{70}-1$), all of the stages to the left of the "most significant" bit of the number contain the sign-bit of the number and indicate whether the number is positive or negative. Thus, the sign-bits ("1's") of a negative number are contained in all of the stages to the left of the left-most zero of the number.

Since the basic arithmetic operation of the 1103 is subtraction, a zero generated during any arithmetic process will be represented by the simultaneous zero state of each stage. Thus the modulus of the one's complement binary number system which can be represented by a 36-stage register is $2^{36}-1$, and as represented by the 72-stage Accumulator, $2^{72}-1$. It should be noted that the machine expressions of these numbers with positive or negative values represent to the computer numbers of a binary system with expressions in the range zero to $2^{36}-2 \pmod{2^{36}-1}$ and zero to $2^{72}-2 \pmod{2^{72}-1}$. Also the machine treatment of binary numbers assumes that the binary point is located to the right of the right-most bit of a number, thus treating each number as an integer.

PX 71871

In summary, the value of an integer I which may be represented by its binary equivalent (in a one's complement system) in a 36-stage register lies in the range

$$1-2^{35} \leq I \leq 2^{35}-1;$$

and in the 72-stage Accumulator, the range is

$$1-2^{71} \leq I \leq 2^{71}-1.$$

Arithmetic operations machine-wise treat numbers (with values in the above ranges) as integers; however, this does not mean that numerical operations are restricted to integers only or to integers in these ranges. A binary number s may be expressed as $s = s_1 2^{s_2}$. If s_1 and s_2 may be expressed as integers with values in the range appropriate for their placement in the computer, s_1 and s_2 may represent in the computer the number s. Numerical operations involving s and t ($t = t_1 2^{t_2}$) are performed machine-wise by the proper arithmetical procedures involving s_1 and t_1, and s_2 and t_2. Fractions may be represented machine-wise by integral values of s_1 and s_2, s_2 being negative. (See Appendix 6A for representation of fractions in the binary number system.)

If s is scaled to its maximum representation by 36 bits such that $2^{35} > |s_1| \geq 2^{34}$, the number s is said to be normalized. Operations involving floating binary point numbers deal with normalized representations of numbers. Floating binary point operations are coded such that a resultant number will be expressed as a floating binary point number.

When an instruction necessitates the transmittal to the Accumulator of a 36-bit integer, the conditions are established during the operation that change the modulus of the integer from $2^{36}-1$ to $2^{72}-1$. This is effected by assuming the existence of 36 bits to the left of the sign-bit. This "72-bit" integer is then transmitted to the Accumulator with the final modular contents of A reflecting the value of the "72-bit" integer according to the nature of the transmitting operation. If it is desired that the mod $2^{36}-1$ value of the 36-bit integer be retained, the operation assumes that each of the simulated 36 left-hand bits has the value of the sign bit. Such an extension of a 36-bit number is designated as a double length extension, D(L), L being the location address of the 36-bit number and (L) being the contents of that address. If it is desired that the value of the machine expression of the 36-bit integer be left undisturbed by the transmitting operation, the value of the sign-bit of (L) is disregarded, and the assumption is made that 36 zeroes exist to the left of the sign-bit of the number. This "72-bit" number is designated as S(L), a split double-length extension.

When one of the above transmissions to the Accumulator is required, the 36-bit integer is first placed in the X-register (by the operations of whichever instruction is being executed) since the only means of information transfer to or from A is via X. Then, according to the instruction being executed, a machine sequence is performed which accomplishes one of the following: adds D(X) to (A), subtracts D(X) from (A), adds S(X) to (A), or subtracts S(X) from (A).

(2) ARITHMETIC OPERATIONS. - The four sequences which are mentioned above are the foundation for almost all the arithmetic operations which the

computer performs. (Exceptions are the use of the shifting facilities of Q and A to effect multiplication by powers of two and the use of control signals to complement the contents of X and to effect a subtraction of the value of one from the contents of the Accumulator without a transmission from X.) In a similar manner, these four arithmetic sequences have a common basic foundation: the fundamental arithmetic property of the computer, namely, the subtractive nature of the Accumulator. Thus a subtraction is reflected in the final modular contents of the Accumulator as the initial contents of the Accumulator minus a double-length extension of the contents of X, i.e.,

$$(A)_f = (A)_i - D(X)$$

or

$$(A)_f = (A)_i - S(X)$$

The operations of the addition sequence result in the final modular contents of the Accumulator reflecting the initial contents of A minus the complement of a double-length extension of the contents of X, i.e.,

$$(A)_f = (A)_i - D(X)'$$

or

$$(A)_f = (A)_i - S(X)'$$

As an example, using a four-bit X-Register and an eight-bit Accumulator, the addition of the following two numbers, $(A)_i$ and (X), is resolved by the addition sequence accordingly:

$(A)_i$ = 0001 0110 (=22)

(X) = 1100 (=-3)

	0001 0110		$(A)_i$
minus	0000 0011	minus	$D(X)'$
	0001 0011		$(A)_f$ (=19)

Any references in this text to the addition of a number to the Accumulator should thus be interpreted as, machine-wise, the process of subtracting the complement of the number from the Accumulator. It should be noted that this basic process of subtraction eliminates, as the result of an arithmetic operation, the occurrence of a one in each stage of the Accumulator as a representation of zero.

Actually, the machine sequence which performs subtraction uses the procedure of the addition sequence by first complementing the desired value content of X_i, and then performing the addition sequence so that the difference resulting is reflected in the Accumulator as

$$(A)_f = (A)_i - D\ (X)_i']\ '$$

or

$$(A)_f = (A)_i - [S(X)_i']\ '$$

As an example, the split subtraction of the following two numbers, $(A)_i$ and (X), is performed by a machine sequence which subtracts S(X) from A according to the following procedure:

PX 71871

$(A)_i$ = 0000 0110 (=6)

(X) = 1100 (=-3)

$S(X)_i$ = 0000 1100 (=12)

$S(X)_i'$ = 1111 0011

$S(X)_i''$ = 0000 1100

```
          0000 0110
minus     0000 1100
          1111 1010
          borrow  1
          1111 1001      (=-6)
```

The process of subtraction necessitates an ability to borrow from a left-hand digit or digits; machine-wise this is made possible by the parallel construction of the stages of the Accumulator. When the value of the machine expression of the subtrahend exceeds the value of the machine expression of the minuend, the subtraction is made possible by the facility of the Accumulator to make an end-around borrow, i.e., a borrow propagated past the stage A_{71} is applied to the stage A_0. The remainder of such a subtraction will thus always have the correct modular value. (See Appendix 6A for further discussion of end-around borrow.)

Multiplication is performed machine-wise by using the shifting facilities of the Accumulator and the Q-Register and the operations which add and subtract D(X) into (A). The execution of an instruction which orders a machine multiplication places the multiplier in the Q-Register, Q, the multiplicand in the X-Register, X, and forms the product in the Accumulator A. The product is formed by adding the multiplicand, or D(X) machine-wise, the appropriate number of times, as determined by the bits of the multiplier, (Q), into the properly positioned Accumulator. The procedure, known as the Multiply Sequence, is as follows:

Repeat 36 times:

1. Shift (A) left one place.
2. Determine the value of the successive digits $(Q_{35} \ldots Q_0)$ beginning with $(Q)_{35}$. If $(Q)_i = 1$, add D(X) to (A).

Thus the multiplication of a number in X by a number in Q results in a sum in the Accumulator of the products of the multiplicand and the coefficients (times their corresponding power of two, $2^{35} \ldots 2^0$) of the digits of the multiplier. The following example, using four-bit Q and X registers and an eight-bit Accumulator illustrates this procedure. In this example the machine process which shift the sums of the products and facilitates machine-wise the multiplication process is not shown; also the sums are not formed (by complementing and subtracting) as they would be machine-wise.

PX 71871

(X) = 0011 multiplicand of 3

(Q) = 0101 multiplier of 5

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		stages of A
0	0	0	0	0	0	0	0		initial contents of A
0	0	0	0	0	0	0	0		D(X) · (Q_3) · 2^3
0	0	0	0	1	1	0	0	plus	D(X) · (Q_2) · 2^2
0	0	0	0	1	1	0	0		
0	0	0	0	0	0	0	0		D(X) · (Q_1) · 2^1
0	0	0	0	0	0	1	1	plus	D(X) · (Q_0) · 2^0
0	0	0	0	1	1	1	1		final contents of A (product = 15)

If the multiplier has a negative value representation, the contents of the Q-Register is represented as $2^{36} - 1 - |(Q)|$ and the product formed in A by the above procedure is

$$(X) \cdot \left[2^{36} - 1 - |(Q)|\right] \text{ or } - |(Q)| \cdot (X) + (X) \cdot 2^{36} - (X).$$

Since the desired value is $- |(Q)| \cdot (X)$, two corrections must be made to eliminate the remaining terms above. The correction $-(X) \cdot 2^{36}$ is applied by subtracting D(X) from (A) previous to step 1 of the Multiply Sequence. The correction + (X) is made by adding D(X) to (A) after the 36th performance of step 2.

An example of multiplication with a multiplier with a negative value representation is given below:

(X) = 0011 multiplicand of 3

(Q) = 1010 multiplier of -5

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		stages of A
0	0	0	0	0	0	0	0		initial contents of A
0	0	1	1	0	0	0	0	minus	D(X) · 2^4
1	1	0	0	1	1	1	1		
0	0	0	1	1	0	0	0	plus	D(X) · (Q_3) · 2^3
1	1	1	0	0	1	1	1		
0	0	0	0	0	0	0	0		D(X) · (Q_2) · 2^2
0	0	0	0	0	1	1	0	plus	D(X) · (Q_1) · 2^1
1	1	1	0	1	1	0	1		
0	0	0	0	0	0	0	0		D(X) · (Q_0) · 2^0
0	0	0	0	0	0	1	1	plus	D(X)
1	1	1	1	0	0	0	0		final contents of A (product = -15)

PX 71871

If the instruction being executed is such that the product of (Q) and (X) is to be added to a number already in the Accumulator, the content of A is shifted 36 places to the left preceding any of the multiply operations. This positions the most significant digits of (A) in A_R in readiness for the additions of the multiplicand to A and the shifting operations. Prior to the actual multiplication operations of the Multiply Sequence, the initial content of A is tested for the condition $2^{71} > |(A)_i| \geq 2^{70}$. If this is evidenced by $(A)_{35} \neq (A)_{34}$, an A Fault is indicated on the Supervisory Control Panel and machine operations are stopped. This condition indicates the possibility of an "overflow" during the Multiply Sequence: i.e., the modular sums resulting from the additions of D(X) to (A) may reach the positive or negative value capacity of A, $2^{71} - 1$ or $1 - 2^{71}$, and as the result of continuing additions, become a number, $S \geq 1 - 2^{71}$ or $S \leq 2^{71} - 1$, thus destroying the cumulative effect desired.

The machine process of division as ordered by the Divide instruction employs the Accumulator, the X-Register, and the Q-Register in such a manner that

$$(A)_i = (X) \cdot (Q) + (A)_f, \quad 0 \leq (A)_f < |(X)| .$$

The 72-bit dividend is initially contained in the Accumulator, the divisor is placed in the X-Register, and the quotient is formed in the Q-Register with the remainder of the division left in A. The division process utilizes the shifting facilities of Q and A and the operations which add and subtract D(X) into A. In general, the process of division, the Divide Sequence, consists of a procedure, repeated 36 times, of inserting a zero or one in the proper stage of Q and adding or subtracting D(X) into A, the operation being determined by a relationship between appropriate digits of the dividend and the divisor.

The basic process of the Divide Sequence is as follows: Decrease a positive dividend by the product of the divisor and descending powers of two, beginning with 2^{35}, until a negative number results; increase this remainder by the product of the divisor and successive descending powers of two until a positive number results; decrease this number, etc. Continue this procedure until the product of the divisor and 2^0 has been added or subtracted, yielding the final remainder. Note that the product mentioned above may in itself be positive or negative depending on the sign of the divisor. Each time a product is subtracted, a one is inserted in the rightmost stage of the Q-Register and subsequently shifted left once; and each time a product is added, the contents of Q as it stands are shifted left once. After the final shifting of those bits as inserted in Q, the register will contain a correct value of the quotient although a negative final remainder as derived above may necessitate an increase or decrease in the value of the quotient. If the final remainder of the above procedure is negative, the remainder is increased by the absolute value of the divisor and the quotient adjusted accordingly by increasing or decreasing its value by one.

The examples below illustrate the basic principle of the Divide Sequence, using four- and eight-bit registers, disregarding the machine processes of shifting the bits in the Accumulator and the Q-Register which facilitate the division machinewise. Also, the additions and subtractions shown are not performed (by complementing and subtracting) as they would be according to their respective machine sequences.

Example 1

Dividend is 0000 1110, $(A)_i$ (=14)
Divisor is 0100, (X) (= 4)

0000 1110	Dividend	= +14
010 0000	minus D(X) • 2^3	- 32
1110 1101		-18
0001 0000	plus D(X) • 2^2	+ 16
1111 1101		- 2
0000 1000	plus D(X) • 2^1	+ 8
0000 0110		+ 6
0000 0100	minus D(X) • 2^0	- 4
0000 0010	Remainder	= + 2

In this example the quotient is derived as follows: the divisor is such that it may be subtracted from the dividend 2^3 -2^2 -2^1 $+2^0$ times, or in binary

1000
-0100
-0010
+0001
0011 (=3)

which is the contents of the Q-Register after the final shifting of the bits inserted into it. Thus the final results of this division are

Quotient is 0011, $(Q)_f$ (=3)
Remainder is 0000 0010, $(A)_f$ (=2)

Example 2

Dividend is 0000 1110, $(A)_i$ (=14)
Divisor is 1100, (X) (=-3)

0000 1110	Dividend	= +14
1110 0111	plus D(X) • 2^3	+(-24)
1111 0101		-10
1111 0011	minus D(X) • 2^2	-(-12)
0000 0010		+ 2
1111 1001	plus D(X) • 2^1	+(- 6)
1111 1011		- 4
1111 1100	minus D(X) • 2^0	-(- 3)
1111 1110	Remainder	= - 1

In this example the quotient is derived as follows: the divisor is such that it may be subtracted from the dividend -2^3 $+2^2$ -2^1 $+2^0$ times, or in binary

PX 71871

$$\begin{array}{r} -1000 \\ +0100 \\ -0010 \\ +0001 \\ \hline 1010 \end{array} \quad (=-5)$$

which is the contents of the Q-Register after the final shifting of the bits inserted into it. Thus the results of the division process thus far are

Quotient is 1010, (Q) $(=-5)$
Remainder is 1111 1110, (A) $(=-1)$

but since the remainder resulting from this division is negative, the value of the divisor is subtracted from it and the quotient is subsequently increased by a value of one. Thus the final results of the division process are a

Quotient of 1011, $(Q)_f$ $(=-4)$
Remainder of 0000 0010, $(A)_f$ $(=+2)$

If the dividend is negative, the procedure for dividing is essentially the same with the dividend being first increased to a positive number, then decreased, etc.

During the division sequence (but before the quotient is adjusted for a negative remainder, if such is the case) machine divide checks are made which determine if the value of the quotient should be an integer which would exceed the capacity, $2^{35} - 1 \leq I \leq 1 - 2^{35}$, of the Q-Register. If such is the case, an A Fault is indicated on the Supervisory Control Panel and the machine operations are stopped.

Since these divide checks are made before any final adjustments to a (negative) remainder and quotient, the division process as illustrated previously, but with a negative dividend, could result in an inaccurate value for the quotient if a negative remainder was left during a division in which the quotient was $2^{35} - 1$ or $1 - 2^{35}$. To take care of such cases so that the division will be stopped by a divide check, an initial correction and an end correction is made during all divisions with negative dividends. These are illustrated by the following example in which the division without the initial correction (which decreases the dividend by the value of the divisor) would result in an inaccurate quotient. The division in the example below would not be carried to completion but would be stopped by the occurrence of the divide check A-Fault as is shown.

Example 3

Dividend is 1110 1001, $(A)_i$ $(=-22)$
Divisor is 1100, (X) $(=-3)$

1110 1001	Dividend	-22
1111 1100	plus D(X), initial correction	+ (-3)
1110 0110		-25
1110 0111	minus D(X) · 2^3	-(-24)
1111 1110		-1
1111 0011	minus D(X) · 2^2	-(-12)
0000 1011		+11

PX 71871

In this division the process would be stopped, after the subtraction of $D(X) \cdot 2^2$ from the previous remainder, since the quotient derived thus far would indicate that the divisor can be subtracted from the dividend $2^3 + 2^2 + (\pm 2^1 \pm 2^0)$ times, the value of which in all cases exceeds the value possible to the quotient.

If this division were performed by the divide sequence without making the initial correction above, it would proceed as follows:

1110 1001	Dividend	-22
1110 0111	minus $D(X) \cdot 2^3$	-(-24)
0000 0010		+ 2
1111 0011	plus $D(X) \cdot 2^2$	+(-12)
1111 0101		-10
1111 1001	minus $D(X) \cdot 2^1$	-(- 6)
1111 1011		- 4
1111 1100	minus $D(X) \cdot 2^0$	-(- 3)
1111 1110		- 1

Thus the division indicates that the divisor can be subtracted $+2^3 -2^2 +2^1 +2^0$ (=7) times from the dividend leaving a remainder of -1. The correction to the quotient for this negative remainder would be

0111	Quotient, (Q)	(=7)
plus 1		
1000	Incorrect value of quotient	(=-7)

Thus the final contents of the Q-Register would be the number above with no indication of the overflow of the modular value allowable to the quotient.

3. REPERTOIRE OF INSTRUCTIONS

a. - The logic of the 1103 system is specified as a two-address logic. This means that two references are provided for the execution of an operation, and an instruction to perform this operation is coded accordingly. The references may be the addresses of operands or other instructions in storage, or they may have a special designation as noted in a subsequent sub-paragraph.

An instruction word in the 1103 consists of 36 bits, $i_{35}...i_0$, with the following composition:

Operation code, O.C.	$i_{35}...i_{30}$, six bits
First execution address, u	$i_{29}...i_{15}$, 15 bits
Second execution address, v	$i_{14}...i_0$, 15 bits

A programmed instruction is coded using octal notation; thus it is represented by octal digits as follows:

O.C.	$(i_{35}...i_{30})$	two octal digits
u	$(i_{29}...i_{15})$	five octal digits representing the bits $u_{14}...u_0$
v	$(i_{14}...i_0)$	five octal digits representing the bits $v_{14}...v_0$

According to the function of the instruction, the portions u and v of the word, represented in octal notation, may be designated as follows:

j	one-digit octal number as represented by the left-most bits of u, u_{14}, u_{13}, u_{12}
n	four-digit octal number as represented by the bits $u_{11}, u_{10}, ..., u_0$
k	number of shifts as represented by the right-most bits of v, $v_6, v_5, ..., v_0$

The functions of the instructions in which j, n, and k occur in place of a storage address will explain their purpose.

Following are other symbols used in the explanation of the functions of the instructions:

(1) Parentheses are used to denote "content(s) of", thus:

(u) = 36-bit word at address u

(Q) = 36-bit word in Q

(A) = 72-bit word in A

(A_R) = 36-bit word in A_R

(A_L) = 36-bit word in A_L

PX 71871

(2) Lower case letters:

q_n is the bit represented by the stage Q_n, $35 \geq n \geq 0$

a_n is the bit represented by the stage A_n, $71 \geq n \geq 0$

(3) A prime denotes a complement, such as (Q)' is the complement of the 36-bit word in Q.

(4) Double length extensions:

D(u) is a 72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all alike and equal to the left-most bit of (u).

S(u) is a 72-bit word whose right-hand 36 bits are (u) and whose left-hand 36 bits are all zeros.

D(Q), D(X), S(Q) and S(X) are similarly defined.

L(Q) (u) is a 72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is given by the bit-by-bit product of the corresponding bits of (u) and (Q).

L(Q)'(v) is a 72-bit word whose left-hand 36 bits are zeros and each of whose right-hand 36 bits is given by the bit-by-bit product of the corresponding bits of (v) and the complement of (Q).

b. The listing of the 1103 repertoire of instructions which follows has the instructions grouped according to a basic similarity in their functions. The instructions are presented with their functions, octal codes, and the alphabetical notations which represent them.

(1) SEQUENCED INSTRUCTIONS.

71uv: MULTIPLY (MPuv): Form in A the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

72uv: MULTIPLY ADD (MAuv): Add the (A) the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

73uv: DIVIDE (DVuv): Divide the 72-bit number in A by (u), putting the quotient in Q, and leaving in A a non-negative remainder R. Then replace (v) by (Q). The quotient and remainder are defined by: $(A)_i = (u) \cdot (Q) + R$, where $0 \leq R < |(u)|$. $(A)_i$ denotes the initial contents of A.

74uv: SCALE FACTOR (SFuv): Replace (A) with D(u) (unless u is A). Then left circular shift (A) by 36 places and continue shifting until $a_{34} \neq a_{35}$. Replace the right-hand 15 bits of (v) with the number of left circular shifts, k, which would be necessary to return (A) to its original position.

75jnw: REPEAT (RPjnw): This instruction calls for the next instruction, which will be called NIuv, to be executed n times, its "u" and "v" addresses being modified or not according to the value of j. Normally n executions are made and the program is continued by the execution of the instruction stored at a fixed address F_1, 00000 or 40001. The procedure is:

1. Replace the right-hand 15 bits of (F_1) with the address w.

2. Execute NIuv, the next instruction in the program n times.

3. If j = 0, do not change u and v.

   If j = 1, add one to v after each execution.

   If j = 2, add one to u after each execution.

   If j = 3, add one to u and v after each execution.

4. On completing n executions, take (F_1) as the next instruction.

5. If the repeated instruction is a jump or stop instruction, the occurrence of a jump or stop terminates the repetition. In addition, if NIuv is a Threshold Jump or an Equality Jump, and the jump to address v occurs, (Q) is replaced by the quantity j, n-r where r is the number of executions that have taken place.

(2) TRANSMISSIVE INSTRUCTIONS.

11uv: TRANSMIT POSITIVE (TPuv): Replace (v) with (u).

12uv: TRANSMIT MAGNITUDE (TMuv): Replace (v) with the absolute magnitude of (u).

13uv: TRANSMIT NEGATIVE (TNuv): Replace (v) with the complement of (u).

15uv: TRANSMIT U-ADDRESS (TUuv): Replace the 15 bits of (v) designated by (v_{29}. . .v_{15}) with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

16uv: TRANSMIT V-ADDRESS (TVuv): Replace the right-hand 15 bits of (v) designated by (v_{14}. . .v_0) with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

35uv: ADD AND TRANSMIT (ATuv): Add D(u) to (A). Then replace (v) with (A_R).

36uv: SUBTRACT AND TRANSMIT (STuv): Subtract D(u) from (A). Then replace (v) with (A_R).

(3) Q-CONTROLLED INSTRUCTIONS.

51uv: Q-CONTROLLED TRANSMIT (QTuv): Form in A the number L(Q) (u). Then replace (v) with (A_R).

52uv: Q-CONTROLLED ADD (QAuv): Add to (A) the number L(Q) (u). Then replace (v) with (A_R).

PX 71871

53uv: Q-CONTROLLED SUBSTITUTE (QSuv): Form in A the quantity L(Q) (u) plus L(Q)' (v). Then replace (v) with (A_R). This effects the replacement of selected bits of (v) with the corresponding bits of (u) in those places corresponding to one's in Q.

(4) REPLACE INSTRUCTIONS.

21uv: REPLACE ADD (RAuv): Form in A the sum of D(u) and D(v). Then replace (u) with (A_R).

23uv: REPLACE SUBTRACT (RSuv): Form in A the difference D(u) minus D(v). Then replace (u) with (A_R).

27uv: CONTROLLED COMPLEMENT (CCuv): Replace (A_R) with (u) leaving (A_L) undisturbed. Then complement those bits of (A_R) that correspond to ones in (v). Then replace (u) with (A_R).

54uk: LEFT SHIFT IN A (LAuk): Replace (A) with D(u). Then left circular shift (A) by k places and replace (u) with (A_R). If u is the address of the Accumulator, the first step is omitted, so that the initial contents of A are shifted.

55uk: LEFT SHIFT IN Q (LQuk): Replace (Q) with (u). Then left circular shift (Q) by k places and replace (u) with (Q).

(5) SPLIT INSTRUCTIONS.

31uk: SPLIT POSITIVE ENTRY (SPuk): Form S(u) in A. Then left circular shift (A) by k places, k being $v_6 \cdot \cdot \cdot v_0$ and $v_{14} \cdot \cdot \cdot v_7$ being zero.

32uk: SPLIT ADD (SAuk): Add S(u) to (A). Then left circular shift (A) by k places, k being $v_6 \cdot \cdot \cdot v_0$ and $v_{14} \cdot \cdot \cdot v_7$ being zero.

33uk: SPLIT NEGATIVE ENTRY (SNuk): Form in A the complement of S(u). Then left circular shift (A) by k places, k being $v_6 \cdot \cdot \cdot v_0$ and $v_{14} \cdot \cdot \cdot v_7$ being zero.

34uk: SPLIT SUBTRACT (SSuk): Subtract S(u) from (A). Then left circular shift (A) by k places, k being $v_6 \cdot \cdot \cdot v_0$ and $v_{14} \cdot \cdot \cdot v_7$ being zero.

(6) TWO-WAY CONDITIONAL JUMP INSTRUCTIONS.

44uv: Q-JUMP (QJuv): If q_{35} is one take (u) as NI; if q_{35} is zero, take (v) as NI. Then, in either case, left circular shift (Q) by one place.

46uv: SIGN JUMP (SJuv): If a_{71} is one, take (u) as NI; if a_{71} is zero, take (v) as NI.

47uv: ZERO JUMP (ZJuv): If (A) is not zero, take (u) as NI; if (A) is zero, take (v) as NI.

(7) ONE-WAY CONDITIONAL JUMP INSTRUCTIONS.

41uv: INDEX JUMP (IJuv): Form in A the difference D(u) minus one. Then if a_{71} is one, continue the present sequence of instructions; if a_{71} is zero, replace (u) with (A_R) and take (v) as NI.

42uv: THRESHOLD JUMP (TJuv): If D(u) is greater than (A), take (v) as NI; if not, continue the present sequence. In either case, leave (A) in its initial state.

43uv: EQUALITY JUMP (EJuv): If D(u) equals (A), take (v) as NI; if not, continue the present sequence. In either case leave (A) in its initial state.

(8) ONE-WAY UNCONDITIONAL JUMP INSTRUCTIONS.

14--: INTERPRET (IP--): Let Y represent the address from which CI was obtained. Replace the right-hand 15 bits of (F_1) with the quantity Y plus 1. Then take (F_2) as NI.

37uv: RETURN JUMP (RJuv): Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (u) with the quantity y plus 1. Then take (v) as NI.

45jv: MANUALLY SELECTIVE JUMP (MJjv): If the number j is 0, take (v) as NI. If j is 1, 2 or 3, and the correspondingly numbered MJ selecting switch is set to "jump", take (v) as NI; if this switch is not set to "jump", continue the present sequence.

(9) MAGNETIC TAPE STORAGE INSTRUCTIONS.

64jv: READ MAGNETIC TAPE (RMjnv): Read n blocks from MT unit j (running forward) to 32n consecutive addresses in MC starting with v.

65jnv: WRITE MAGNETIC TAPE (WMjnv): From 32n consecutive addresses in MC, starting with v, write n blocks on MT unit j (running forward).

66jn-: ADVANCE MAGNETIC TAPE (AMjn-): Move the magnetic tape of MT unit j in the forward direction by n blocks.

67jn-: BACK MAGNETIC TAPE (BMjn-): Move the magnetic tape of MT unit j in the backward direction by n blocks.

(10) STOP INSTRUCTIONS.

56jv: MANUALLY SELECTIVE STOP (MSjv): If j is 0, stop computer operation, providing suitable indication. If j is 1, 2, or 3 and the correspondingly numbered MS selecting switch is set to "stop", stop computer operation, providing suitable indication. Whether or not a stop occurs, (v) is NI.

57--: FINAL STOP (FS--): Stop computer operation, providing suitable indication.

Note: For the "suitable indication" provided in instructions MSjv and FS--, see subparagraph d.(1), paragraph 5.

(11) EXTERNAL EQUIPMENT INSTRUCTIONS.

17-v: EXTERNAL FUNCTIONS (EF-v): As indicated by (v) select a unit of external equipment and cause it to perform the designated function.

61-v: PRINT (PR-v): Replace (TWR) with the right-hand six bits of (v). Cause the typewriter to perform the operation specified by the 6-bit code.

63jv: PUNCH (PUjv): Replace (HPR) with the right-hand six bits of (v). Cause the punch to respond to (HPR). If j = 0, omit seventh level hole; if j = 1, include seventh level hole.

76jv: EXTERNAL READ (ERjv): If j =0, replace (v) with (IOA) in $v_7 \ldots v_0$ and zeroes in $v_{35} \ldots v_8$; if j = 1, replace (v) with (IOB).

77jv: EXTERNAL WRITE (EWjv): If j = 0, replace (IOA) with the right-hand eight bits of (v), if j = 1, replace (IOB) with (v). Cause the previously selected unit to respond to the information in IOA or IOB.

c. In the following pages the instructions are presented with their representative octal operation codes in numerical order. The purpose of this presentation is to give the programmer a simplified representation of the detailed operations which the 1103 performs during, and necessary to, the execution of each instruction. The events listed under each instruction occur as a result of the control section receiving and sensing the operation code of the instruction. These events do not necessarily designate a machine operation but represent the actions taken by the machine in executing an instruction. The order in which these events are listed is in accordance, for the most part, with the time-wise sequence of machine operations which they represent. In some instructions the desire for a simplified presentation overruled the desire for a sequential presentation if such was not necessary for accuracy in determining the contents of the various registers at any time during the execution of the instruction. The deviations from machine operation sequences in any of the presentations do not cause a misrepresentation of the actual operations of the machine.

The columns in which the events are listed differentiate them in accordance with their primary function in the execution of the instruction. An event listed in the Procurement of Operands column brings an operand out of storage (MC, MD, Q or A) into the X-Register in preparation for some operation upon it or placement in another register. An event listed in the Operations column may describe arithmetic and/or logical machine operations or may be the clearance of a register in preparation for such operations. An event listed in the third column, Storage of Results, indicates the final placement of a result of actions described in the first and/or second columns.

The symbols and abbreviations used in the presentation of the instructions have been listed previously except for those listed below. The definitions of the following notations do not explain machine operations but the effect of machine operations.

Clear A Replace the contents of each stage of the Accumulator with a zero.

Clear X	Replace the contents of each stage of the X-Register with a zero.
Complement PAK	Replace the contents of each stage of PAK with its complement.
Complement X	Replace the contents of each stage of X with its complement.
$\left.\begin{matrix}u\\v\end{matrix}\right\} \rightarrow$ PAK	Replace the contents of the 15 stages of the Program Address Counter with the MC or MD storage address as designated by u or v.
$F_1 \rightarrow$ PAK	A special case of the above with the MC address specified as being octal 00000, (or MD address 40001).
jn $\rightarrow$ PAK	Replace the contents of the 15 stages of PAK with the bits designated as jn.
$(X_{14}...X_0) \rightarrow$ PAK	Replace the contents of the 15 stages of PAK with the contents of the right-most stages of the X-Register.
PAK $\rightarrow$ X	Replace the contents of the 15 right-most stages of the X-Register, $(X_{14}...X_0)$, with the address held in PAK.
w $\rightarrow$ X	Replace the contents of the 15 right-most stages of the X-Register, $(X_{14}...X_0)$, with the address designated as w.
k $\rightarrow$ X	Replace the contents of the right-most stages of the X-Register with the shift count k.
(IOA) $\rightarrow$ X	Replace the contents of the eight right-most stages of X with (IOA).
(IOB) $\rightarrow$ X	Replace the contents of the 36 stages of X with (IOB).
(PAK) $\rightarrow$ X	Replace the contents of the 36 stages of X with the contents of the address held in PAK.
(u) $\rightarrow$ X	If u is an MC or MD address, replace the contents of the X-Register with the contents of the MC or MD location specified by u. If u is an address of the Q-Register, replace the contents of the X-Register with the contents of the Q-Register. If u is an address of the Accumulator, replace the contents of the X-Register with the contents of A_R unless otherwise noted.
(v) $\rightarrow$ X	If v is the address of an MC or MD location or Q or A, replacement operations occur as above with v being the transmitting register.
(X) $\rightarrow$ IOA	Replace the contents of the eight stages of IOA with the contents of the right-most eight stages of the X-Register.
(X) $\rightarrow$ IOB	Replace the contents of the 36 stages of IOB with the contents of the X-Register.
(X) $\rightarrow$ PCR	Replace the contents of the 36 stages of PCR with the contents of the X-Register.
(X) $\rightarrow$ u	If u is an MC or MD address, replace the contents of the MC or MD location specified by u with the contents of the X-Register. If u is an address of the Q-Register, replace the contents of Q with the contents of the X-Register. If u is an address of the Accumulator, replacement operations

PX 71871

do not occur except in instructions 11, 12, 13, 55, 73, and 76. In these instructions, the replacement is performed by Clear A and Add D(X) to A. A replacement operation (X)→ A occurring in the instructions not listed above would effect undesirable alterations in the contents of A.

$(X) \rightarrow v$

If v is the address of an MC or MD location or Q or A, replacement operations occur as above with v being the receiving register.

$(Xn_2 \ldots Xn_1) \rightarrow \begin{cases} u \\ v \end{cases}$

If the receiving register has an MC or MD address, replace with the contents of the stages $Xn_2 \ldots Xn_1$ only the contents of the corresponding stages of the receiving register, leaving the remaining stages undisturbed. A partial replacement of Q or A is not permissible.

$(X_{14} \ldots X_0) \rightarrow F_1$

A special case of the above with the stages of X being specified as being the right-most 15 and the MC address specified as being 00000 (or MD address 40001).

The contents of the transmitting register are not disturbed in any of the above replacement operations.

PX 71871

Instruction Reference Events

The events listed below conclude the execution of all program instructions, except the Repeat instruction, when they are not immediately preceded by the Repeat instruction. Also, a number of instructions which are preceded by the Repeat instruction are terminated by these events. (See Exceptions, p. 6-72) It is to be understood that these events follow in sequential order the events listed for each appropriate instruction. As a result, the execution of the current instruction is terminated, and the computer is prepared for the execution of the next instruction whose address is acquired from PAK, the Program Address Counter. This address is determined according to the function of the current instruction if this function indicates that the normal sequence of instructions be interrupted. The contents of the location whose address is held in PAK is placed in PCR, the Program Control Register, where it is interpreted as the next instruction to be executed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(PAK) → X Advance PAK by one (X) → PCR	

PX 71871

OPERATION CODE: 11

INSTRUCTION: Transmit Positive, TPuv

FUNCTION: Replace (v) with (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X		(X) → v If v is A, Clear A Add D(X) to (A)

PX 71871

OPERATION CODE: 12

INSTRUCTION: Transmit Absolute Magnitude, TMuv

FUNCTION: Replace (v) with the absolute magnitude of (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Complement (X) if negative	(X) → v If v is A, Clear A Add D(X) to (A)

OPERATION CODE: 13

INSTRUCTION: Transmit Negative, TNuv

FUNCTION: Replace (v) with the complement of (u)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Complement (X)	(X) → v If v is A, Clear A Add D(X) to (A)

PX 71871

OPERATION CODE: 14

INSTRUCTION: Interpret, IP

FUNCTION: Let Y represent the address from which CI was obtained. Replace the right-hand 15 bits of (F_1) with the quantity Y + 1. Then take (F_2) as the next instruction.

F_1 is storage address 00000, F_2 is storage address 00001.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X PAK (i.e., Y + 1) → X Set PAK to F_2	$(X_{14} \ldots X_0) \rightarrow F_1$

PX 71871

OPERATION CODE: 15

INSTRUCTION: Transmit U-Address, TUuv

FUNCTION: Replace the 15 bits of (v) designated by $(v_{29} \ldots v_{15})$ with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$		$(X_{29} \cdots X_{15}) \rightarrow v$ v = Q or A not permissible

PX 71871

OPERATION CODE: 16

INSTRUCTION: **Transmit V-Address, TVuv**

FUNCTION: **Replace the right-hand 15 bits of (v) designated by (v_{14} . . . v_0) with the corresponding bits of (u), leaving the remaining 21 bits of (v) undisturbed.**

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X		(X_{14} ... X_0) → v v = Q or A not permissible

PX 71871

OPERATION CODE: 17

INSTRUCTION: External Function, EF-v

FUNCTION: As indicated by (v) select a unit of external equipment and cause it to perform the designated function.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(v) → X	If previous operations involving (IOB) are completed - (X) → IOB Select, and cause an action of external equipment, according to (IOB)	

PX 71871

OPERATION CODE: 21

INSTRUCTION: Replace Add, RAuv

FUNCTION: Form in A the sum of D(u) and D(v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Clear A	
	Add D(X) to (A)	
(v) → X		
	Add D(X) to (A)	
	(A_R) → X	
		(X) → u Omit if u is A

PX 71871

OPERATION CODE: 23

INSTRUCTION: Replace Subtract, RSuv

FUNCTION: Form in A the difference D(u) minus D(v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Clear A	
	Add D(X) to (A)	
(v) → X		
	Subtract D(X) from (A)	
	(A_R) → X	
		(X) → u Omit if u is A

PX 71871

OPERATION CODE: 27

INSTRUCTION: Controlled Complement, CCuv

FUNCTION: Replace (A_R) with (u) leaving (A_L) undisturbed. Then complement those bits of (A_R) that correspond to ones in (v). Then replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X		
	Clear A_R	
	Complement (X)	
	Complement (A_{35} ... A_0) if corresponding (X_{35} ... X_0) is zero	
(v) → X		
	Complement (X)	
	Complement (A_{35} ... A_0) if corresponding (X_{35} ... X_0) is zero	
	(A_R) → X	
		(X) → u Omit if u is A

PX 71871

OPERATION CODE: 31

INSTRUCTION: Split Positive Entry, SPuk

FUNCTION: Form S(u) in A. Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Clear A Add S(X) to (A) Shift (A) left k places	

If the bits $v_{14} \ldots v_7$ of instruction 31 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows:

(If instruction 31 is at address y, PAK will currently contain address y + 1.)

$r_0 = PAK_0$
.
.
.
$r_6 = PAK_6$
$r_7 = PAK_7 + v_7$
.
.
.
$r_{14} = PAK_{14} + v_{14}$

} bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

PX 71871

OPERATION CODE: 32

INSTRUCTION: Split Add, SAuk

FUNCTION: Add S(u) to (A). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Add S(X) to (A) Shift (A) left k places	

If the bits $v_{14} \ldots v_7$ of instruction 32 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows:

(If instruction 32 is at address y, PAK will currently contain address y + 1.)

$r_0 = PAK_0$
.
.
.
$r_6 = PAK_6$

$r_7 = PAK_7 + v_7$
.
.
.
$r_{14} = PAK_{14} + v_{14}$

} bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

PX 71871

OPERATION CODE: 33

INSTRUCTION: Split Negative Entry, SNuk

FUNCTION: Form in A the complement of S(u). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Clear A Subtract S(X) from (A) Shift (A) left by k places	

If the bits $v_{14} \ldots v_7$ of instruction 33 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows:

(If instruction 33 is at address y, PAK will currently contain address y + 1.)

$r_0 = PAK_0$
. .
. .
. .
$r_6 = PAK_6$
$r_7 = PAK_7 + v_7$
. .
. .
. .
$r_{14} = PAK_{14} + v_{14}$

bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

PX 71871

OPERATION CODE: 34

INSTRUCTION: Split Subtract, SSuk

FUNCTION: Subtract S(u) from (A). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$ and $v_{14} \ldots v_7$ being zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Subtract S(X) from (A) Shift (A) left by k places	

If the bits $v_{14} \ldots v_7$ of instruction 34 are not "0's", the next instruction to be executed will be taken from address r with the bits of this address, $r_{14} \ldots r_0$, being determined as follows:

(If instruction 34 is at address y, PAK will currently contain address y + 1.)

$r_0 = PAK_0$
. .
. .
. .
$r_6 = PAK_6$
$r_7 = PAK_7 + v_7$
. .
. .
. .
$r_{14} = PAK_{14} + v_{14}$

} bit-by-bit sum with the exception that a PAK_i of "1" and a v_i of "1" will result in an r_i of "1".

After the execution of the instruction at address r, the next instruction to be executed will be taken from address y + 2 (unless the instruction at r called for a jump).

PX 71871

OPERATION CODE: 35

INSTRUCTION: Add and Transmit, ATuv

FUNCTION: Add D(u) to (A). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Add D(X) to (A) $(A_R) \rightarrow X$	$(X) \rightarrow v$ Omit if v is A

PX 71871

OPERATION CODE: 36

INSTRUCTION: **Subtract and Transmit, STuv**

FUNCTION: **Subtract D(u) from (A). Then replace (v) with (A_R).**

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Subtract D(X) from (A) (A_R) → X	(X) → v Omit if v is A

PX 71871

OPERATION CODE: 37

INSTRUCTION: Return Jump, RJuv

FUNCTION: Let y represent the address from which CI was obtained. Replace the right-hand 15 bits of (u) with the quantity y plus one. Then take (v) as the next instruction. (See Notes, p. 6-52)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X PAK (i.e., y + 1) → X v → PAK	$(X_{14} \cdots X_0) \rightarrow u$ u = Q or A not permissible

PX 71871

OPERATION CODE: 41

INSTRUCTION: **Index Jump, IJuv**

FUNCTION: **Form in A the difference D(u) minus one. Then if a_{71} is one, continue the present sequence of instructions; if a_{71} is zero, replace (u) with (A_R) and take (v) as NI.** **(See Notes, p. 6-52)**

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
(u) → X Omit if u is A		
	Clear A Omit if u is A	
	Add D(X) to (A)	
	Subtract one from (A)	
	(A_R) → X	
	v → PAK	
if a_{71} is zero		(X) → u Omit if u is A

PX 71871

OPERATION CODE: 42

INSTRUCTION: **Threshold Jump, TJuv, not repeated**

FUNCTION: Subtract (u) from (A). If a_{71} is then one, take (v) as the NI; if a_{71} is zero, continue the present sequence of instructions. Then in either case, restore (A) to its initial state. (See Notes, p.6-52)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Subtract D(X) from (A) v → PAK if a_{71} is one Add D(X) to (A)	

PX 71871

OPERATION CODE: 42

INSTRUCTION: Threshold Jump, TJuv, repeated

FUNCTION: Subtract (u) from (A). If a_{71} is then one, replace (Q) with j,n-r and take (v) as the next instruction; if a_{71} is zero, continue with the present sequence of instructions. In either case, restore (A) to its initial state.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Subtract D(X) from (A) If a_{71} is zero, Add D(X) to (A) Advance (PAK) Test (PAK) for condition indicating n repeats and proceed accordingly (See Note 2). If a_{71} is one, Complement (PAK) Add D(X) to (A) Perform jump termination (See Note 2)	

Note: 1. This instruction is preceded by instruction 75jnw which leaves the complement of "jn" in PAK.

2. See Jump Termination, page 6-73.

PX 71871

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv, not repeated

FUNCTION: Subtract (u) from (A). If (A) is then zero, take (v) as the next instruction; if (A) is not zero, continue the present sequence. In either case, restore (A) to its initial state. (See Notes, page 6-52.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Subtract D(X) from (A) Subtract one from (A) v → PAK if end-borrow was propagated from A_{71} Add D(X) to (A) Set (X) to 1 Add D(X) to (A)	

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv, repeated

FUNCTION: Subtract (u) from (A). If (A) is then zero, replace (Q) with j, n-r and take (v) as next instruction; if (A) is not zero, continue the present sequence. In either case, restore (A) to its initial state.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Subtract D(X) from (A) Subtract one from (A) If no end-borrow was propagated from A_{71} Add D(X) to (A) Set (X) to one Add D(X) to (A) Advance PAK Test (PAK) for condition indicating n repeats and proceed accordingly. (See Note 2.) If end-borrow was propagated from A_{71} Complement (PAK) Add D(X) to (A) Set (X) to one Add D(X) to (A) Perform jump termination (See Note 2.)	

Note 1. This instruction is preceded by instruction 75jnw which leaves the complement of "jn" in PAK.

Note 2. See Jump Termination, page 6-73.

OPERATION CODE: 44

INSTRUCTION: Q-Jump, QJuv

FUNCTION: If q_{35} is one, take (u) as NI; if q_{35} is zero, take (v) as NI. Then, in either case, left circular shift (Q) by one place. (See Notes, p.6-52)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	If q_{35} is one, u → PAK If q_{35} is zero, v → PAK Shift (Q) left one place	

OPERATION CODE: 45

INSTRUCTION: Manually Selective Jump, MJjv

FUNCTION: If the number j is 0, take (v) as NI. If j is 1, 2, or 3, and the correspondingly numbered MJ Selecting Switch is set to "jump", take (v) as the NI; if this switch is not set to "jump", continue the present sequence. (See Notes, p.6-52)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	For jump, v → PAK	

PX 71871

OPERATION CODE: 46

INSTRUCTION: Sign Jump, SJuv

FUNCTION: If a_{71} is one, take (u) as NI; if a_{71} is zero, take (v) as NI. (See Notes, p. 6-52)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	If a_{71} is zero, v → PAK If a_{71} is one, u → PAK	

OPERATION CODE: 47

INSTRUCTION: Zero Jump, ZJuv

FUNCTION: If (A) is not zero, take (u) as NI; if (A) is zero, take (v) as NI. (See Notes, page 6-52.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Subtract one from (A) v → PAK if end-borrow was propagated from A_{71} u → PAK if no end-borrow was propagated from A_{71} Set (X) to one Add D(X) to (A)	

Notes Concerning the Jump Instructions

When the conditions are such that the contents of the v (or u) address are referred to as the next instruction to be executed, the following notes are applicable:

1. Machinewise, v = A is not permissible.

2. If v = Q, the next instruction to be executed after the termination of the jump instruction will be (Q). If (Q) is a legal instruction it will be executed in the normal manner. Its execution will be repeated unless it is a jump instruction since the address of each succeeding instruction is taken from PAK, and PAK will advance from 10000 to 11777 (and revert back to 10000) unless a FORCE stop or a jump is called for.

3. The above remarks also apply to u for the two-way jump instructions.

OPERATION CODE: 51

INSTRUCTION: Q-Controlled Transmit, QTuv

FUNCTION: Form in A the number L(Q)(u). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Clear A Form in X bit-by-bit product of (Q) and (X) Add S(X) to (A)	(X) → v Omit if v is A

PX 71871

OPERATION CODE: 52

INSTRUCTION: Q-Controlled Add, QAuv

FUNCTION: Add to (A) the number L(Q)(u). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
$(u) \rightarrow X$	Form in X bit-by-bit product of (Q) and (X) Add S(X) to (A) $(A_R) \rightarrow X$	$(X) \rightarrow v$ Omit if v is A

PX 71871

OPERATION CODE: 53

INSTRUCTION: Q-Controlled Substitute, QSuv

FUNCTION: Form in A the quantity L(Q)(u) plus L(Q)'(v). Then replace (v) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Clear A Form in X bit-by-bit product of (Q) and (X) Add S(X) to (A) Complement (Q)	
(v) → X	Form in X bit-by-bit product of (Q) and (X) Add S(X) to (A) Complement (Q) (A_R) → X	(X) → v Omit if v is A

PX 71871

OPERATION CODE: 54

INSTRUCTION: Left Shift in A, LAuk

FUNCTION: Replace (A) with D(u). Then left circular shift (A) by k places, k being $v_6 \ldots v_0$, and replace (u) with (A_R).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X Omit if u is A	Clear X Clear A Omit if u is A Add D(X) to (A) Shift (A) left k places (A_R) → X	(X) → u* Omit if u is A

*The 54 (and 55) instruction does not always store the shifted number at the address from which the number, before shifting, was obtained. The address to which the $(A_R)_f$ will be transmitted is determined in this manner: If the 15 bits of the receiving address of the shifted number are designated as $r_{14} \ldots r_0$, these bits are determined as follows:

$r_0 = u_0$
. .
. .
. .
$r_6 = u_6$

$r_7 = u_7 + v_7$
. . .
. . .
. . .
$r_{14} = u_{14} + v_{14}$

} bit-by-bit sum with the exception that a u_i of "1" and a v_i of "1" will result in an r_i of "1"

Therefore, by choosing $v_{14} \ldots v_7$ with care, (u) may be left undisturbed. A particular advantage of this peculiarity is that the shifted number may be placed in Q or left in A without designating Q or A as the u-address.

PX 71871

OPERATION CODE: 55

INSTRUCTION: Left Shift in Q, LQuk

FUNCTION: Replace (Q) with (u). Then left circular shift (Q) by k places, k being $v_6 \ldots v_0$, and replace (u) with (Q).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	(X) → Q Shift (Q) left k places (Q) → X	(X) → u* If u is A, Clear A Add D(X) to (A)

*The 55 (and 54) instruction does not always store the shifted number at the address from which the number, before shifting, was obtained. The address to which the $(A_R)_f$ will be transmitted is determined in this manner: If the 15 bits of the receiving address of the shifted number are designated as $r_{14} \ldots r_0$, these bits are determined as follows:

$$r_0 = u_0$$
$$\vdots$$
$$r_6 = u_6$$
$$r_7 = u_7 + v_7$$
$$\vdots$$
$$r_{14} = u_{14} + v_{14}$$

bit-by-bit sum with the exception that a u_i of "1" and a v_i of "1" will result in an r_i of "1".

Therefore, by choosing $v_{14} \ldots v_7$ with care, (u) may be left undisturbed. A particular advantage of this peculiarity is that the shifted number may be placed in Q or left in A without designating Q or A as the u-address.

PX 71871

OPERATION CODE: 56

INSTRUCTION: Manually Selective Stop, MSjv

FUNCTION: If j is 0, stop the computer operation, providing suitable indication. If j is 1, 2, or 3 and the correspondingly numbered MS selecting switch is set to "stop", stop computer operation, providing suitable indication. Whether or not a stop occurs, (v) is NI.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	v → PAK Stop computer operation if stop indicated.	

Note: v = A not permissible; for v = Q, see Note 2, p. 6-52.

OPERATION CODE: 57

INSTRUCTION: Final Stop, FS--

FUNCTION: Stop computer operation, providing suitable indication, (See sub-paragraph d.(1), Paragraph 6)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Stop computer operation.	

OPERATION CODE: 61

INSTRUCTION: Print, PR-v

FUNCTION: Replace (TWR) with the right-hand six bits of (v). Cause the typewriter to perform the operation specified by the 6-bit code.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(v) → X If TWR is clear, place one's in ($TWR_5 \ldots TWR_0$) where they are present in corresponding ($X_5 \ldots X_0$) Perform typewriter operation as specified by (TWR).	

PX 71871

OPERATION CODE: 63

INSTRUCTION: Punch, PUjv

FUNCTION: Replace (HPR) with the right-hand six bits of (v). Cause the punch to respond to (HPR). If j = 0, omit seventh level hole; if j = 1 include seventh level hole.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(v) → X If HPR is clear, place a one in HPR_6 if j = 1 and place ones in ($HPR_5 \ldots HPR_0$) where they are present in corresponding ($X_5 \ldots X_0$) Punch (HPR)	

PX 71871

OPERATION CODE: 64

INSTRUCTION: Read Magnetic Tape, RMjnv

FUNCTION: Read n blocks (see Note) from MT unit j (running forward) to 32n consecutive addresses in MC starting with v.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Repeat 32n times; Insert in Q successive words from MT unit j. (Q)→X (X)→v + 0...32n	

Note: One block of magnetic tape consists of 32 words, each of 36 bits.

PX 71871

OPERATION CODE: 65

INSTRUCTION: Write Magnetic Tape, WMjnv

FUNCTION: From 32n consecutive addresses in MC, starting with v, write n blocks (see Note) on MT unit j (running forward).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Repeat 32n times: (v + 0...32n) → X (X) → Q Write (Q) on MT unit j until n successive blocks have been written.	

Note: One block of magnetic tape consists of 32 words, each of 36 bits.

PX 71871

OPERATION CODE: 66

INSTRUCTION: Advance Magnetic Tape, AMjn-

FUNCTION: Move the magnetic tape of MT unit j in the forward direction by n blocks (see Note).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Advance MT unit j n blocks.	

Note: One block of magnetic tape consists of 32 words, each of 36 bits.

PX 71871

OPERATION CODE: 67

INSTRUCTION: Back Magnetic Tape, BMjn-

FUNCTION: Move the magnetic tape of MT unit j in a backward direction by n blocks (see Note).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Back MT unit j n blocks.	

Note: One block of magnetic tape consists of 32 words, each of 36 bits.

OPERATION CODE: 71

INSTRUCTION: Multiply, MPuv

FUNCTION: Form in A the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X		
	Clear A	
	(X) → Q	
(v) → X		
	Form in A the product of (Q) and (X)	

PX 71871

OPERATION CODE: 72

INSTRUCTION: Multiply Add, MAuv

FUNCTION: Add to (A) the 72-bit product of (u) and (v), leaving in Q the multiplier (u).

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X		
	(X) → Q	
	Shift (A) left 36 places	
(v) → X		
	Add to (A) shifted the product of (Q) and (X)	

PX 71871

OPERATION CODE: 73

INSTRUCTION: Divide, DVuv

FUNCTION: Divide the 72-bit number in A by (u), putting the quotient in Q, and leaving in A a non-negative remainder, R. Then replace (v) by (Q). The quotient and remainder are defined by: $(A)_i = (u) \cdot (Q) + R$ where $0 \leq R < |(u)|$. $(A)_i$ denotes the initial contents of A.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
(u) → X	Clear Q Divide (A) by (X), placing the quotient in Q and leaving R in A (Q) → X	(X) → v If v is A, Clear A Add D(X) to (A)

PX 71871

OPERATION CODE: 74

INSTRUCTION: Scale Factor, SFuv

FUNCTION: Replace (A) with D(u) unless u is A. Then left circular shift (A) 36 places and continue shifting until $a_{35} \neq a_{34}$. Replace the right-hand 15 bits of (v) with the number of left shifts, k, necessary to return the final contents of A, or $(A)_f$, to the original position. The range of k if u is A is $0 \leq k \leq 71$: if u is MC, MD, or Q, k may be 0 or $37 \leq k \leq 71$. Effectively, the initial contents of A, or $(A)_i$, which may be D(u) or D(Q) after the above replacement, are positioned in A_R (with the sign bit represented by A_{35} and the most significant bit by A_{34}) so that $(A)_f = (A)_i \cdot 2^s$. If $0 \leq k \leq 36$, the Scale Factor $s = -k$; if $37 \leq k \leq 71$, $s = 72 - k$. Note that for $0 < k \leq 36$, this positioning scales $(A)_i$ "down"; for $37 < k \leq 71$, $(A)_i$ are scaled "up". If $k = 0$, $(A)_i$ were properly positioned before any shifting operations; if $k = 37$, $(A)_i$ are all ones or zero.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X	
(u) → X Omit if u is A		
	Clear A Omit if u is A	
	Add D(X) to A	
	Shift (A) left 36 places and continue shifting until $a_{35} \neq a_{34}$	
	Clear X	
	k → X	
		$(X_{14} \ldots X_0) \rightarrow v$ v = Q or A not permissible

PX 71871

OPERATION CODE: 75

INSTRUCTION: Repeat, RPjnw

FUNCTION: This instruction calls for the next instruction, NIuv, to be executed n times, $0 \leq n \leq 2^{12}-1$, its "u" and "v" addresses being modified or not according to the value of j. Normally n executions are made, and the program is continued by the execution of the instruction stored at a fixed address F_1. The instruction usually stored at F_1 is MJjv, calling for a jump to its v-address, which according to the workings of the Repeat instruction means that (w) of RPjnw will be taken as the next instruction.

The u and v addresses of the repeated instruction are advanced according to the value of j as follows:

If j = 0, neither the "u" nor the "v" execution address of the repeated instruction is advanced.
j = 1, the "v" execution address of the repeated instruction is advanced after each execution.
j = 2, the "u" execution address of the repeated instruction is advanced after each execution.
j = 3, both the "u" and "v" execution addresses of the repeated instruction are advanced after each execution.

During a repeat sequence, PAK is used as a counter to record the number of times the instruction is executed.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X w → X (PAK) → X jn → PAK Complement PAK (X) → PCR Begin Repeat Sequence: Advance (PAK) and investigate its contents. If n = 0, proceed to Normal Repeat Termination If n ≠ 0, execute NI n times,* advancing u and v addresses according to j	$(X_{14} \ldots X_0) \rightarrow F_1$

*See Exceptions, p. 6-72.

Intermediate Steps following each execution of an instruction whose repeat is possible.

FUNCTION: Test for n executions of repeated instruction; proceed to Normal Repeat Termination if n executions have occurred: if not, repeat instruction.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Advance PAK by one Test PAK for condition indicating n executions of repeated instruction and proceed accordingly.	

Normal Repeat Termination

FUNCTION: Terminate the repeat sequence when an instruction has been executed n times by taking (F_1) as the next instruction. (See Exceptions)

	(F_1) $\longrightarrow$ X (X) $\longrightarrow$ PCR	

Abnormalities:

1. If the fixed address F_1 is not a Jump instruction, the address of the instruction to be executed following the instruction stored in F_1 is determined by the modified complement of "j" from the Repeat instruction. This is the number that remains stored in PAK at the end of a repeat sequence that has a Normal Termination. If, in 75jnw, "j" was 0, the address will be 40000; if "j" was 1, the address will be 70000; if "j" was 2, the address will be 60000; if "j" was 3, the address will be 50000.

2. In addition to values j = 0, 1, 2, and 3, the value "j" in 75jnw may also be 4, 5, 6, or 7. An instruction following a Repeat instruction where j = 4, 5, 6, or 7 will be repeated indefinitely unless its execution produces a jump or stop operation or a jump termination. If these latter conditions do not occur, the "u" and "v" execution addresses of the repeated instruction will be advanced depending on "j": if j = 4, neither address is advanced; if j = 5, the "v" address is advanced; if j = 6, the "u" address is advanced; if j = 7, both addresses are advanced. If "u" or "v" are MD addresses, they may be advanced from 40000 through 77777, and the next advance signal after 77777 will start the address sequence over from 40000. If "u" or "v" are MC, Q, or A addresses, they may be advanced 1023 addresses from their respective first addresses, and the next advance signal will start their address sequence over from their respective first addresses.

3. If a Repeat instruction is immediately followed by a second Repeat instruction, the second will supersede the first, and the address of the repeated instruction is determined by the number stored in PAK. This address is the complement of "jn-1" which remains stored in PAK from the first Repeat instruction.

TERMINATION OF A REPEAT SEQUENCE

The first step of the repeat sequence is to determine whether the value of "n" in the Repeat instruction is zero or not. If "n" is zero (indicating that no execution of the instruction following the Repeat instruction is to be made) the attempted repeat sequence is immediately terminated according to the Normal Repeat Termination. If $n \neq 0$, the instruction, regardless of what it is, is executed. The repeat sequence is concluded by a Normal Repeat Termination, after n executions, except when the instruction following RPjnw is one of the following:

Exceptions:

(1) If the Interpret (14--), Return Jump (37uv), Q Jump (44uv), Sign Jump (46uv), Zero Jump (47uv), Manually Selected Stop (56jv), or Final Stop (57--) instruction is to be executed after a Repeat instruction, the attempted repeat sequence is automatically terminated. These instructions are thus executed as if no Repeat instruction preceded them and terminated by the normal Instruction Reference Events.

(2) If either the Index Jump (41uv) or Manually Selected Jump (45jv) is to be executed following a Repeat instruction, they may be repeated "n" times as specified if no jump operation is called for in the course of their execution. If a jump operation is called for, the instruction is terminated according to the normal Instruction Reference Events and the content of the jump address is taken for the NI. (No count of the number of times the instruction was executed is retained, however.) If no jump operation is called for in "n" executions of these instructions, a Normal Repeat Termination is executed and (F_1) are taken for the NI.

(3) If either the Threshold Jump (42uv) or the Equality Jump (43uv) is to be executed following a Repeat instruction, they can be repeated or not depending on whether or not in their execution the threshold or equality conditions are reached and a jump operation is called for. If a jump operation is called for before or exactly after "n" executions, the repeat sequence of these instructions is terminated by a special Jump Termination (see p.6-73). The Jump Termination stores the quantity j,n-r in Q and takes the contents of the jump address for the NI. The (Q) can then be used to determine how many times the instruction was executed. If "n" executions of these instructions are performed and the threshold or equality conditions requiring a jump do not occur, the repeat sequence of these instructions is terminated by the Normal Repeat Termination and (F_1) are taken for the NI.

PX 71871

Jump Termination for repeated instructions 42uv and 43uv.

FUNCTION: When a jump condition, as specified by one of the instructions above, is reached before or immediately after n executions, the steps listed below, followed by the normal Instruction Reference Events, terminate the execution of the instruction.

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(PAK) → X (X) → Q v → PAK	

PX 71871

OPERATION CODE: 76

INSTRUCTION: External Read, ERjv

FUNCTION: If j = 0, replace (v) with (IOA) in $v_7 \ldots v_0$ and zeroes in $v_{35} \ldots v_8$; if j = 1, replace (v) with (IOB). If the external unit utilizes step-by-step operation, advance one step. (This instruction must be preceded by the External Functions instruction.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	Clear X If IOA or IOB has received information from external equipment (IOA) or (IOB) → X Clear IOA or IOB	(X) → v If v is A, Clear A Add D(X) to A

If IOA is involved, note that when v is a 36-bit register, the eight right-most bits of v are (IOA) and the left-most 28 bits are zeroes; when v is A, the eight right-most bits of A are (IOA) and the remaining 64 bits are zeroes.

PX 71871

OPERATION CODE: 77

INSTRUCTION: External Write, EWjv

FUNCTION: If j = 0, replace (IOA) with the right-hand eight bits of (v); if j = 1, replace (IOB) with (v). Cause the previously selected unit to respond to the information in IOA or IOB. (This instruction must be preceded by the External Functions instruction.)

PROCUREMENT OF OPERANDS	OPERATIONS	STORAGE OF RESULTS
	(v) → X If previous operations involving (IOA) or (IOB) are completed - (X) → IOA or IOB Transmit (IOA) or (IOB) to external equipment	

PX 71871

d. INSTRUCTION EXECUTION TIMES. - On the following pages the instructions are presented with accompanying tables which give the time required for the execution of each instruction, the execution being repeated or not repeated. The time required for the execution of an instruction is dependent upon the storage references, where such are made, by the u and/or v addresses. The tables for the instructions with such references applicable and practical list individually the execution times of each instruction with its u and/or v addresses referencing an MC register, the Accumulator, or the Q-Register. (All execution times assume that the instructions are stored in MC.) The specified times are measured, unless otherwise stated, from MPO to MPO: each number in a table is the number of clock pulse periods, (two microseconds each), required for the execution of a particular coded instruction.

Included in the determination of the execution time for an instruction repeated n times are the times required by the Repeat instruction and, in most cases, the instruction stored at F_1. This total execution time is given by the sum, $29+R_n+p$, where R_n is the execution time of the instruction repeated n times and p is the execution time of the instruction stored at F_1. If n is zero, R_n is zero and, in all cases, the total execution time is 29 + p and the next instruction is taken from F_1.

When certain of the instructions are repeated, the total execution time does not include the time p. These instructions are as follows: Interpret 14--; Return Jump, 37uv; Q-Jump, 44uv; Sign Jump, 46uv; Zero Jump, 47uv; Manually Selective Stop, 56uv; and Final Stop, 57--. These instructions are executed only once at the most regardless of an n>1. The next instruction to be executed will not be taken from F_1.

Other exceptions to including the time p in the total execution time are made under certain conditions when the following instructions are repeated: Index Jump, 41uv; Manually Selective Jump, 45jv; Threshold Jump, 42uv; and Equality Jump, 43uv. The time p is not added when, during the repeated n executions of these instructions, the jump requirement is met before or during the nth execution of the instruction. The occurrence of the jump eliminates the procedure of taking the next instruction to be executed from F_1.

Execution times for repeating certain of the instructions are not given if a repeat of such instructions has no practical use.

The following notations are used in the tables of execution times:

SCC	Storage Class Control indicates the coded u and/or v addresses being used are not permissible.
-	Execution time not given for a possible instruction at a Q address.
N	Number of executions of the repeated instruction.
(u_i)	Denotes bit in stage i, $(35 \geq i \geq 0)$, of register addressed as u.
$(u_i)_j$	Contents of u_i during repeat j, j = 1 ... N.
ms	Milliseconds.

PX 71871

OPERATION CODE: 11

INSTRUCTION: Transmit Positive, TPuv

NON-REPEATED

u \ v	MC	A	Q
MC	20	19	17
A	16	15	13
Q	17	16	14

REPEATED

u \ v	MC
MC	13N
A	9N
Q	10N

OPERATION CODE: 12

INSTRUCTION: Transmit Magnitude, TMuv

NON-REPEATED

u \ v	MC	A	Q
MC	21	20	18
A	17	16	14
Q	18	17	15

REPEATED

u \ v	MC
MC	14N
A	10N
Q	11N

PX 71871

OPERATION CODE: 13

INSTRUCTION: Transmit Negative, TNuv

NON-REPEATED

u \ v	MC	A	Q
MC	20	19	17
A	16	15	13
Q	17	16	14

REPEATED

u \ v	MC
MC	13N
A	9N
Q	10N

OPERATION CODE: 14

INSTRUCTION: Interpret, IP--

NON-REPEATED

u and v irrelevant
17

INSTRUCTION: Transmit U-Address, TUuv
Transmit V-Address, TVuv

OPERATION CODE: 15
16

NON-REPEATED

u \ v	MC	A	Q
MC	20	SCC	SCC
A	16	SCC	SCC
Q	17	SCC	SCC

REPEATED

u \ v	MC	A	Q
MC	13N	SCC	SCC
A	9N		
Q	10N		

INSTRUCTION: External Function, EF-v

OPERATION CODE: 17

NON-REPEATED

v	MC	A	Q
	16	12	13

computer operating time, not including a lockout time if IOB is currently in use for output operations.

INSTRUCTION: Replace Add, RAuv

OPERATION CODE: 21

NON-REPEATED

u \ v	MC	A	Q
MC	33	29	30
A	25	21	22
Q	27	23	24

REPEATED

u \ v	MC	A	Q
MC	26N	22N	23N
A	19N		
Q	20N		

INSTRUCTION: Replace Subtract, RSuv

OPERATION CODE: 23

NON-REPEATED

u \ v	MC	A	Q
MC	34	30	31
A	26	22	23
Q	28	24	25

REPEATED

u \ v	MC	A	Q
MC	27N	23N	24N
A	20N		
Q	21N		

OPERATION CODE: 27

INSTRUCTION: Controlled Complement, CCuv

NON-REPEATED

u \ v	MC	A	Q
MC	29	25	26
A	21	17	18
Q	23	19	20

REPEATED

u \ v	MC	A	Q
MC	22N	18N	19N
A	15N		
Q	16N		

OPERATION CODE: 31
32

INSTRUCTION: Split Positive Entry, SPuk
Split Add, SAuk

NON-REPEATED

u	
MC	18 + k
A	14 + k
Q	15 + k

REPEATED

u	
MC	(11 + k) N

where k is not altered by the Repeat Sequence. For k=0 and 1 use value of k=2.

PX 71871

OPERATION CODE: 33
34

INSTRUCTION: Split Negative Entry, SNuk
Split Subtract, SSuk

NON-REPEATED

u	
MC	19 + k
A	15 + k
Q	16 + k

REPEATED

u	
MC	(12 + k) N

where k is not altered by the Repeat Sequence. For k=0 and 1, use value of k=2.

OPERATION CODE: 35

INSTRUCTION: Add and Transmit, ATuv

NON-REPEATED

u \ v	MC	A	Q
MC	24	20	21
A	20	16	17
Q	21	17	18

REPEATED

u \ v	MC	A	Q
MC	17N	14N	14N
A	13N		
Q	14N		

PX 71871

OPERATION CODE: 36

INSTRUCTION: Subtract and Transmit, STuv

NON-REPEAT

u \ v	MC	A	Q
MC	25	21	22
A	21	17	18
Q	22	18	19

REPEATED

u \ v	MC	A	Q
MC	18N	15N	15N
A	14N		
Q	15N		

OPERATION CODE: 37

INSTRUCTION: Return Jump, RJuv

NON-REPEATED

u \ v	MC	A	Q
MC	19	SCC	--
A	SCC	SCC	SCC
Q	SCC	SCC	SCC

REPEATED*

u \ v	MC	A	Q
MC	12	SCC	--
A	SCC	SCC	SCC
Q	SCC	SCC	SCC

*See preliminary discussion

OPERATION CODE: 41

INSTRUCTION: Index Jump, IJuv

NON-REPEATED

u \ v	MC	A	Q
MC	29	SCC	-
A	21	SCC	-
Q	23	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	19r+3	SCC	-
A	15	SCC	-
Q	16	SCC	-

where r = number of executions up to the occurrence of the jump.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	24
A	20
Q	21

REPEATED

u \ v	irrelevant
MC	19N
A	15N
Q	16N

NO JUMP

*See preliminary discussion

PX 71871

OPERATION CODE: 42

INSTRUCTION: Threshold Jump, TJuv

NON-REPEATED

u \ v	MC	A	Q
MC	23	SCC	-
A	19	SCC	-
Q	20	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	16r+5	SCC	-
A	17	SCC	-
Q	18	SCC	-

where r = number of executions up to the occurrence of the jump. The jump will occur during the first execution if u is A or Q.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	23
A	19
Q	20

REPEATED

u \ v	irrelevant
MC	16N
A	12N
Q	13N

NO JUMP

*See preliminary discussion

OPERATION CODE: 43

INSTRUCTION: Equality Jump, EJuv

NON-REPEATED

u \ v	MC	A	Q
MC	29	SCC	-
A	25	SCC	-
Q	26	SCC	-

REPEATED*

u \ v	MC	A	Q
MC	22r+5	SCC	-
A	23	SCC	-
Q	24	SCC	-

where r = number of executions up to the occurrence of the jump. The jump will occur during the first execution if u is A or Q.

JUMP*

NON-REPEATED

u \ v	irrelevant
MC	29
A	25
Q	26

REPEATED

u \ v	irrelevant
MC	22N
A	18N
Q	19N

NO JUMP

*See preliminary discussion

OPERATION CODE: 44
46

INSTRUCTION: Q-Jump, QJuv
Sign Jump, SJuv

NON-REPEATED

u or v	
MC	10
A	SCC
Q	-

REPEATED*

u or v	
MC	3
A	SCC
Q	-

OPERATION CODE: 45

INSTRUCTION: Manual Jump, MJjv

NON-REPEATED

v	MC	A	Q
	10	SCC	-

JUMP

v	irrelevant
	10

NO JUMP

REPEATED*

v	MC	A	Q
	3	SCC	-

JUMP*

v	irrelevant
	5N

NO JUMP

*See preliminary discussion

OPERATION CODE: 47

INSTRUCTION: Zero Jump, ZJuv

NON-REPEATED

u or v	
MC	16
A	SCC
Q	-

REPEATED*

u or v	
MC	9
A	SCC
Q	-

OPERATION CODE: 51

INSTRUCTION: Q-Controlled Transmit, QTuv

NON-REPEATED

u \ v	MC	A	Q
MC	24	20	21
A	20	16	17
Q	21	17	18

REPEATED

u \ v	MC	A	Q
MC	17N	14N	14N
A	13N		
Q	14N		

*See preliminary discussion

OPERATION CODE: 52

INSTRUCTION: Q-Controlled Add, QAuv

NON-REPEATED

u \ v	MC	A	Q
MC	25	21	22
A	21	17	18
Q	22	18	19

REPEATED

u \ v	MC	A	Q
MC	18N	15N	15N
A	14N		
Q	15N		

OPERATION CODE: 53

INSTRUCTION: Q-Controlled Substitute, QSuv

NON-REPEATED

u \ v	MC	A	Q
MC	40	32	34
A	36	28	30
Q	37	29	31

REPEATED

u \ v	MC	A	Q
MC	33N	26N	27N
A	29N		
Q	30N		

PX 71871

OPERATION CODE: 54

INSTRUCTION: Left Shift in A, LAuk

NON-REPEATED

u \ v*	MC	A	Q
MC	24 + k	20 + k	21 + k
A	16 + k	16 + k	SCC
Q	18 + k	SCC	18 + k

REPEATED

u \ v*	MC
MC	(17 + k)N

where k is not altered by the Repeat Sequence.

*v is coded with bits $v_6 \ldots v_0$ representing k and bits $v_{14} \ldots v_7$ completing an MC, A, or Q address. If v is an A address, (u) shifted remain in A; if v is a Q address, the final (A_R) are transmitted to Q.

OPERATION CODE: 55

INSTRUCTION: Left Shift in Q, LQuk

NON-REPEATED

u \ v*	MC	A	Q
MC	23 + k	22 + k	20 + k
A	18 + k	18 + k	SCC
Q	17 + k	SCC	17 + k

REPEATED

u \ v*	MC
MC	(16 + k)N

Where k is not altered by the Repeat Sequence.

*Where v is coded with bits $v_6 \ldots v_0$ representing k and bits $v_{14} \ldots v_7$ completing an MC, A, or Q address. If v is an A address, (u) shifted are transmitted to A; if v is a Q address, (u) shifted remain in Q.

PX 71871

OPERATION CODE: 56

INSTRUCTION: Manually Selective Stop, MSjv

NON-REPEATED

v	MC	A	Q
	10	SCC	-

REPEATED*

v	MC	A	Q
	3	SCC	-

NO STOP

v	irrelevant
	2

v	irrelevant
	-5

STOP

OPERATION CODE: 57

INSTRUCTION: Final Stop, FS--

NON-REPEATED

u and v irrelevant

1

REPEATED*

u and v irrelevant

-6

*See preliminary discussion

OPERATION CODE: 61
63

INSTRUCTION: Print, PR-v
Punch, PUjv

NON-REPEATED

v	MC	A	Q
	19	15	16

Computer operating time, not including a possible lockout time due to external equipment cycle times of - approximately 105ms for typewriter,
16.7 ms for Punch

(Cycle times listed are not maximum lockout times)

REPEATED

v	MC	A	Q
	14N	10N	11N

Computer operating time, not including lockout times due to external equipment cycle times.

Approximate overall time in ms:
Maximum -
Punch 16.7 (N-1) + 12.5
Typewriter 105N
Minimum -
Punch 16.7 (N-1)
Typewriter 105(N-1)

OPERATION CODE: 64
65

INSTRUCTION: Read Magnetic Tape, RMjnv
Write Magnetic Tape, WMjnv

NON-REPEATED

v	MC
	(14+144n) ms

$n \neq 0$

v	MC
	10ms

n=0

REPEATED

v	MC
	(14+144n) N ms

$n \neq 0$

v	MC
	10N ms

n=0

PX 71871

OPERATION CODE: 66
67

INSTRUCTION: Advance Magnetic Tape, AMjn -
Back Magnetic Tape, BMjn -

NON-REPEATED	REPEATED
u and v irrelevant	u and v irrelevant
14	8N
computer operating time, exclusive of a (maximum) lockout time if the same MT unit is referred to of (14+144n) ms	computer operating time, exclusive of computer lockout time because the same MT unit is referred to of(14+144n) (N-1) ms.

$n \neq 0$

u and v irrelevant	u and v irrelevant
14	8N
computer operating time, exclusive of a (maximum) lockout time if the same MT unit is referred to of 10 ms.	computer operating time, exclusive of computer lockout time because the same MT unit is referred to of 10(N-1) ms.

$n = 0$

PX 71871

OPERATION CODE: 71

INSTRUCTION: Multiply, MPuv

NON-REPEATED

u \ v	MC	A	Q
MC	61	57	58
A	57	53	54
Q	58	54	55

each plus $2(u_0) + 4\sum_{i=1}^{35}(u_i) + 7(u_{35})$

REPEATED

Subtract 36 from corresponding execution times of Multiply Add instruction.

OPERATION CODE: 72

INSTRUCTION: Multiply Add, MAuv

NON-REPEATED

Add 36 to corresponding execution times of Multiply instruction.

REPEATED

u \ v	MC	A	Q
MC	90N	86N	87N
A	86N	82N	83N
Q	87N	83N	84N

Minimum times where $(u_{35})_j$ and $(u_0)_j$ are zeroes and the additional time required for $(u_{34} \ldots u_1)_j$ of ones is given by the summation below the table.

each plus $4 \sum_{j=1}^{N} \left(\sum_{i=1}^{34} u_i \right)_j$

u \ v	MC	A	Q
MC	239N	235N	234N
A	235N	231N	232N
Q	236N	232N	233N

Maximum times where $(u_{35} \ldots u_0)_j$ are all ones.

If the contents of $(u_{35} \ldots u_0)_j$ are known:

u = MC, v = MC

$$90\,N + \sum_{j=1}^{N} \left\{ 2(u_0)_j + 7(u_{35})_j + 4 \sum_{i=1}^{35} (u_i)_j \right\}$$

u = Q, v = Q

$$84N + N \left\{ 2(q_0) + 7(q_{35}) + 4 \sum_{i=1}^{35} (q_i) \right\}$$

PX 71871

OPERATION CODE: 73

INSTRUCTION: Divide, DVuv

NON-REPEATED

u \ v	MC	A	Q
MC	243	242	240
A	239	238	236
Q	240	239	237

add to each $4(a_{71})$

REPEATED

u \ v	MC
MC	$236\ N + 4 \sum_{j=1}^{N} (a_{71})_j$

Maximum time (in case of a preliminary negative remainder)

NON-REPEATED

u \ v	MC	A	Q
MC	240	239	237
A	236	235	233
Q	237	236	234

add to each $4(a_{71})$

REPEATED

u \ v	MC
MC	$233N + 4 \sum_{j=1}^{N} (a_{71})_j$

Minimum time

PX 71871

OPERATION CODE: 74

INSTRUCTION: Scale Factor, SFuv

NON-REPEATED

u	
MC	63 + γ
A	59 + γ
Q	60 + γ

REPEATED

If v=A or Q, SCC fault

u	
MC	56 + γ

where $\gamma = (36-k) \bmod 72$ and k is the scale factor $0 \leq k \leq 71$. For k=37, use value of k=38.

OPERATION CODE: 75

INSTRUCTION:

NON-REPEATED*

$29 + R_n + p$

where p is the execution time of the terminal jump at F_1 and R_n is the execution time of the repeated instruction. (If n=0, R_n=0)

REPEATED

If the RP instruction is repeated, the second one takes precedence over the first.

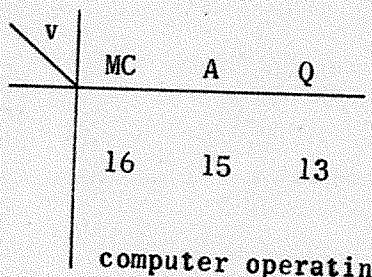
*See preliminary discussion

PX 71071

OPERATION CODE: 76

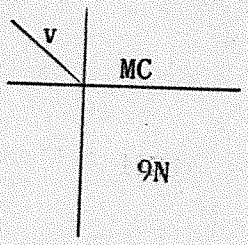
INSTRUCTION: External Read, ERjv

NON-REPEATED



computer operating time, not including a lockout time if IOB, or IOA, has not yet received the information being "read".

REPEATED

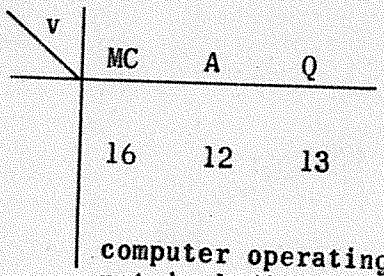


computer operating time, not including lockout times.

OPERATION CODE: 77

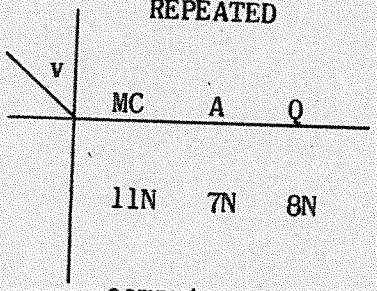
INSTRUCTION: External Write EWjv

NON-REPEATED



computer operating time, not including a lockout time if IOB, or IOA, is currently in use for output operations.

REPEATED



computer operating time, not including lockout times.

PX 71871