RPC-4000 programming manual

HLT	INP	PRU	TBC
SNS	EXC	EXT	STU
CXE	DVU	MML	STL
RAU	DIV	CME	CLU
RAL	SHIFT	*TEMPORARY EDITION*	CLL
SAU	SLC	**april 1960**	ADU
MST	MPY		ADL
LDC	MPT	CMG	SBU
LDX	PRD	TMI	SBL

FOREWORD

THIS TEMPORARY EDITION OF THE RPC PROGRAMMING MANUAL IS BEING PUBLISHED TO PROVIDE MATERIAL RELATIVE TO THE COMMAND REPERTOIRE AND MODES OF OPERATION OF THE RPC-4000 FOR THOSE PERSONS WITH AN IMMEDIATE NEED FOR THIS INFORMATION.

THE MANUAL, AS PRESENTED, IS INCOMPLETE, CONSISTING OF FOUR OF AN EVENTUAL SEVEN SECTIONS. THE COMPLETE MANUAL, TO BE PUBLISHED IN THE NEAR FUTURE, WILL CONTAIN SECTIONS ON PROGRAMMING TECHNIQUES, MANUAL CONTROLS AND I/O DEVICES, AND SUMMARY LISTS AND TABLES.

AN INTRODUCTION TO COMPUTERS

1

THE EMERGENCE OF THE ELECTRONIC COMPUTER AS A MAJOR TOOL FOR BUSINESS AND INDUSTRY IS NOT SURPRISING WHEN IT IS CONSIDERED THAT A VERY LARGE PART OF TODAY'S TOTAL WORK EFFORT IS DEVOTED TO THE PROCESSING OF LENGTHY CALCULATIONS OR VAST AMOUNTS OF STATISTICAL DATA. MUCH OF THIS COMPUTATION INVOLVES WEARISOME REPETITION ALONG WITH THE NECESSITY TO EFFECTIVELY UTILIZE A CONFUSING ARRAY OF INTERRELATED INFORMATION. IT IS JUST THIS SORT OF JOB THAT CAN BE HANDLED BEST BY AN INTELLIGENTLY DIRECTED MACHINE. NOT ONLY CAN THE COMPUTER PERFORM THESE TASKS MANY TIMES FASTER, BUT WITH MUCH LESS FALLIBILITY THAN CAN THE HUMAN WORKER.

SINCE EARLIEST TIMES, THE DEVELOPMENT OF COMPUTING AIDS HAS FOLLOWED TWO SEPERATE PATHS, DEPENDING UPON THE MEANS EMPLOYED TO RECOGNIZE AND TO REPRESENT INFORMATION VALUES. THOSE DEVICES WHICH OPERATE BY THE MEASUREMENT OF CONTINUOUS PHYSICAL VARIABLES ARE KNOWN AS ANALOG DEVICES OR ANALOG COMPUTERS. A FUEL GAUGE IS A SIMPLE ANALOG DEVICE IN THAT THE DEFLECTION OF A POINTER IS ANALOGOUS TO THE QUANTITY OF FUEL IN A TANK.

THE OTHER TYPE OF COMPUTING AID, AND THAT WHICH DIRECTLY CONCERNS US, IS KNOWN AS A DIGITAL COMPUTER. IT IS CHARACTERIZED BY ITS REPRESENTATION OF VALUES, BOTH QUANTITATIVE AND SYMBOLIC, BY COUNTS OF DISCRETE DISCONTINUOUS PHYSICAL UNITS. THE ABACUS IS A SIMPLE DIGITAL DEVICE IN WHICH BEADS ARE USED TO REPRESENT THE COUNTING UNITS. A MORE COMPLEX AND MODERN, BUT STILL NON-AUTOMATIC DIGITAL DEVICE IS REPRESENTED BY THE DESK CALCULATOR.

THE AUTOMATIC DIGITAL COMPUTER

THE ABACUS HAS SERVED MAN FOR THOUSANDS OF YEARS AS A SIMPLE STATIC STORAGE DEVICE TO HOLD THE PROGRESSIVE RESULTS OF LENGTHY CALCULATIONS. THE MODERN DESK CALCULATOR HAS, IN ADDITION TO A SIMILAR LIMITED STORAGE CAPACITY, THE ABILITY TO MECHANICALLY PERFORM CERTAIN BASIC ARITHMETIC OPERATIONS, SUCH AS ADDITION, DIVISION, ETC. HOWEVER, BOTH OF THESE AIDS TO COMPUTATION REQUIRE THE CONSTANT SERVICES OF THE HUMAN OPERATOR TO DIRECT EACH INDIVIDUAL OPERATION. THUS, THE SPEED OF THESE NON-AUTOMATIC DEVICES IS LIMITED TO THE RAPIDITY WITH WHICH THE OPERATOR CAN CONTROL THEIR ACTIONS. IN ORDER TO FUNCTION AUTOMATICALLY, A COMPUTER MUST BE SELF-SEQUENCING; THAT IS, IT MUST HAVE THE CAPABILITY OF CONTROLLING THE ORDER IN WHICH THE STEPS OF A CALCULATION ARE PERFORMED, BY REFERENCE TO A SERIES OF CODED SIGNALS WHICH ARE STORED WITHIN ITSELF.

The history of the Automatic Digital Computer goes back to the year 1822, when an Englishman by the name of Charles Babbage, with the financial backing of the British Government, began work on what he called a full size "Difference Engine". This machine was designed for the purpose of calculating mathematical tables so as to relieve the human operator of this routine function. Ten years later, after an expenditure of 17,000 pounds, the project was abandoned. In 1833 he elaborated upon his initial efforts, to develop the concept of a universal computer which he called an "Analytical Engine". It was designed to be fully automatic and externally programmed. It incorporated facilities for input/output, arithmetic operations, internal storage and automatic program control. Unfortunately, the state of the engineering art was insufficiently advanced to produce a machine of such mechanical complexity. Although a considerable amount of money and effort were expended, the machine was never completed.

It was not until the year 1944 that the first such universal digital computer was actually completed. This machine, generally referred to as the Harvard Mark I, used electromagnetic relays and mechanical counters, and was extremely cumbersome compared with the computers in use today. The first computer to substitute electronic circuitry for electromagnetic was the ENIAC (Electronic Numerical Integrator And Calculator), which was used primarily to solve ballistics problems. It contained some 18,000 vacuum tubes in addition to about 1500 electro-mechanical relays.

The growth rate of the computer industry following these pioneer efforts has been truly fantastic. Digital Computers are being entrusted with an ever increasing share of the routine, repetitive functions of business and Industry. Computers are being produced in a variety of types and size classes depending on their intended use. Of particular importance is the class of small to medium size computers being applied to such diverse problems as process control, data reduction for management analysis, inventory control and scientific problem solving. They are characterized by moderate cost, ease of installation and maintenance, and great flexibility in their application to a variety of tasks.

Computer Design Considerations

An automatic Digital Computer must, of necessity, contain certain basic logical elements. It must, of course, have the ability to perform a number of simple arithmetic and logical operations. The solution of most mathematical equations, regardless of complexity, can be reduced to a series of basic arithmetic operations. Thus, the ability to add, subtract, multiply and divide is sufficient to perform virtually any mathematical computation. These operations must, however, be performed in a specifically directed manner in order that the desired data manipulation may be accomplished. The circuitry which interprets the instruction codes, brings in the proper operands, and effects the step-by-step performance of a required operation is generally referred to as the Arithmetic Element. It is to this element that are assigned the tasks of processing and relaying information and of elementary decision making.

In order that information (both instructions and operands) may be supplied to the Arithmetic Element at a rate comparable to its inherent processing speed, it is necessary that there be some form of internal Memory Element. This element is generally used to store both the program to direct the processing operation, and the data to be processed. It must provide for rapid access and locatability of each piece of information on a program demand basis. Without the Memory Element, the speed of a computer would be restricted to a mechanical insertion rate.

Prior to beginning a processing operation, and often during the course of an operation, information from an external source must be entered into the computer's Memory. Consequently, there is required an Input Element to accomplish this loading of data.

Conversely, at the completion of, and often during an operation, the processed data must be externally presented in a usable and understandable manner. This internal to external transfer of data is a function of the Output Element.

Finally, and vital to the automatic functioning of the computer, there must be a Control Element to coordinate the activities of the other four elements. It is responsible for the proper sequencing of each action within the computer. It is this precise sequencing which determines whether a piece of information stored in Memory is to be used as an instruction or as data to be processed. It is the only way in which the computer can make this distinction. Data and instructions occur intermixed in Memory, and in the same form. In fact, the identical piece of information may be used in both ways in different parts of a program. Further, this Control Element must activate Input and Output devices, when called for, and, in general, direct the overall performance of a program or program complex.

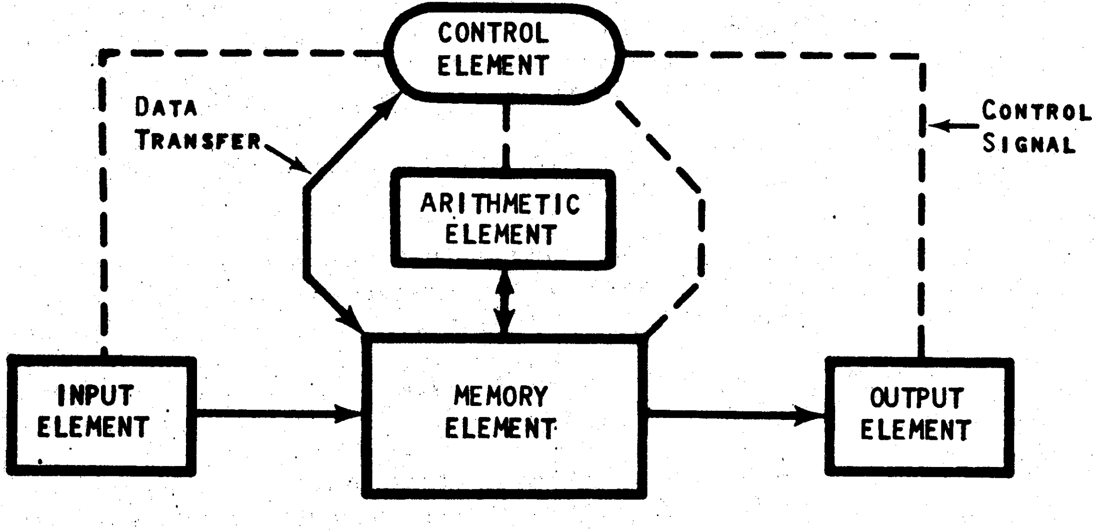


FIGURE 1. Logical Diagram of Automatic Computer

GLOSSARY OF COMPUTER TERMS

ACCESS TIME----The time required to bring a selected Word from storage to the point at which it is to be used or processed.

ACCUMULATOR----That Register, within the Arithmetic Element, in which are formed the results of arithmetical and logical operations.

ADDRESS--------A character, or string of characters, used to identify a location within the Computer Memory.

ADDRESS (Absolute)--------A label, usually numeric, which identifies a location in Memory relative to the initial Memory location.

ADDRESS (Symbolic)--------A label, consisting of arbitrarily chosen symbols to represent a location within a Program Routine which is independent of the location of the Routine in Memory or its initial location.

ALPHA-NUMERIC--A symbol system in which are included alphabetic, numeric and special characters.

ARITHMETIC ELEMENT-----That part of a computer in which arithmetic and logical computations and decision making functions are performed.

ASSEMBLER------A Utility Program which assigns Absolute Address values for the values in a Symbolically Addressed Program Routine and sets up storage allocations for its various parts.

BASE-----------In a number system employing positional notation, the Base is the number of counts required in each position to cause a change in the next higher position. It is also the number of discrete numeric characters employed in the system.

BINARY-CODED DECIMAL---Representation of each decimal character in a number by a pattern of Binary digits.

BIT------------Common programming expression for Binary Digit. The smallest meaningful unit of information in the Computer. An individual Bit is restricted to the values "0" and "1".

BLOCK----------A group of related Computer Words or characters processed or transferred as a unit.

BOOTSTRAP------A procedure for entering a Program into the Computer. The initial few steps of the Routine, normally entered manually, are used to automatically load the remainder of the Program.

BUFFER---------An intermediate storage device for coordinating the transfer of information from one part of the computer to another.

COMMAND--------The directive portion of an Instruction. The specified action to be taken by the Computer.

COMPILER-------A Utility Program which produces a machine language Program from a Program which is coded in a problem oriented language. The coding form of the Program to be compiled ordinarily will closely approximate standard algebraic notation.

COMPUTER (Digital)-------An electronic device for the automatic calculation of sequences of arithmetic and logical operations. Quantities and values are represented by patterns of bi-stable magnetic or electronic indicators.

CONSTANT-------A value which is not subject to change during an operation.

CONTROL ELEMENT--------That part of a Computer which directs the sequencing and timing of its actions.

DATA-----------That information used as Operands in the arithmetic and logical operations of the Computer.

DATA REDUCTION-The processing of large volumes of raw Data so as to condense and simplify it to a more meaningful presentation.

DEBUGGING------The process of eliminating errors from a Program by inspection or machine testing.

EXTRACT--------To clear selected portions of a Word to zero, leaving the remaining portions intact.

FIELD----------A defined space within a Computer Word or information format which is assigned to hold a specified type or class of information.

FIXED POINT----That system of programming arithmetic in which the location of the decimal or binary point in a Computer Word must be controlled and manipulated by the programmer.

FLOATING POINT-That system of programming arithmetic in which the location of the decimal or binary point in a Computer Word is automatically manipulated and controlled by the Computer or a Program Routine.

HEAD-----------The assembly for recording or reading one Track of information on a magnetized surface.

HEXADECIMAL----The positional number system using a Base of 16. A number system which employs 16 discrete numeric characters.

INDEX REGISTER-A Register to contain a quantity which may be used to automatically increment the Address portion of an Instruction.

INPUT----------Information entered into a Computer's Memory from an external source.

INSTRUCTION----A set or string of characters which completely specifies one action to be taken by the Computer.

LOAD-----------To enter information into the Computer from an external source. Also, to place a value into a Register, such as the Index Register.

LOOP-----------A Programming technique in which a sequence of Instructions is repeated a specified number of times before proceeding with the remainder of the Program.

MAGNETIC DRUM--A rotating cylindrical drum used for information storage. Recording is in the form of magnetized spots on the surface of the drum.

MEMORY---------The working storage area of a Computer. Generally, the storage device permitting the most rapid access to its data. It is from this storage that Instructions are obtained for execution.

MICROSECOND----One millionth of a second.

MILLISECOND----One thousandth of a second.

MNEMONIC-------A code form of identification devised so as to assist in the remembrance of its meaning.

OPERAND--------An item of information which is to be operated upon, or one which enters into an operation.

OPTIMIZE-------To code a Routine in such a way as to minimize the total Memory Access Time.

OUTPUT---------Information transferred from the Computer's Memory to an external device.

OVERFLOW-------The generation, in a Computer Register, of a quantity beyond the capacity of the Register.

PARAMETER------An item which may be assigned arbitrary values, depending upon its use in a given Routine.

PARITY BIT-----An extra Binary Digit added to an item of information for validity checking purposes.

PARITY CHECK---A method of verifying the accuracy of a data transfer by counting the number of "1" Bits in the transferred item, including the Parity Bit. An accurate transfer is indicated by an even count in an "even parity" system, or by an odd count in an "odd parity" system.

PROGRAM--------A complete sequenced set of Computer Instructions designed to carry out a desired processing function, or solve a defined problem.

REGISTER-------The hardware for storing one complete Computer Word.

ROUTINE--------A sequenced set of Computer Instructions, part of a Program, for performing some well defined function.

SECTOR---------The space on a storage drum, measured along the circumference, required to hold one Computer Word.

SUBROUTINE (Closed)-----A Program sub-unit, usually used in common by more than one Program, which is entered via a Transfer from the main Program and exits via a Transfer back to a selected point in the main Program.

SUBROUTINE (Open)-----A Program sub-unit, not entered via a Transfer, which is included in the normal sequence of a Program.

TRACK----------The path around a storage drum, traced out by each Head of the drum assembly. Also, the information stored on a given Track.

TRANSFER-------To move information from one storage area to another.

To depart from the linear sequence of the Program Instructions by shifting control to another area of the Program. This shift is often conditional upon the results of a Program test of an indicator Word. (Conditional Transfer)

UTILITY PROGRAM--------A Program designed to assist in the full utilization of the Computer. Included in this class are Assembly Programs, Compilers, Input/Output Programs, etc.

WORD-----------A set or string of symbols which occupies one complete storage register in the Computer. This Word may be treated as an Instruction or as a Data Word, depending upon the manner of its occurrence in a Program.

2 ARITHMETIC FOR PROGRAMMERS

The concept of number is so basic to our everyday life that the average person rarely has occasion to reflect on the many meanings of number, or on the many forms in which numbers are expressed. However, the advent of the electronic computer has necessitated a reconsideration of number systems, and their applicability to the conveyance and manipulation of information through electronic circuitry.

Numbers are used, primarily, to denote quantity or amount. We use numbers to state how many pills are in a bottle, or how many pounds of coffee are in a bag, or how many miles between here and Tin Cup, Colorado. But there are other ways of using numbers. A number can also be used to represent one member of an ordered set of symbols. A house number, for example, serves to identify a particular house among a number of houses in its set. It further serves to locate that house with respect to these other houses. When a number is used for this dual purpose of location and identification, it is generally referred to as an address. Finally, a number which serves only to identify, but which has no quantitative or locational significance, is usually called a code number.

All three of these aspects of number---quantity, address and identity---are vital to the design and operation of a high speed digital computer. However, the present discussion will concern itself, primarily, with the quantitative aspect of number.

NUMBER SYSTEMS

A Number System may be classified by the number of counting symbols it employs. This number is referred to as the base of the system. The Decimal System is a base-ten Number System; that is, it uses the ten numeric characters, 0 thru 9. It is further characterized by its use of a positional notation. When counting, if one digit position progresses beyond 9, it adds a count in the position immediately to its left. Thus, 9 plus 1 becomes 10; 19 plus 1 becomes 20; 99 plus 1 becomes 100, etc. Each position in a decimal number represents an integral power of 10, so that in the number 456, the 4 represents 4 times 10^2; the 5 represents 5 times 10^1; and the 6 represents 6 times 10^0. Similarly, the decimal fraction .789 represents 7 times 10^{-1} plus 8 times 10^{-2} plus 9 times 10^{-3}.

Obviously, we are not restricted to a base-ten system of numbers. The same notational structure applies equally well to other bases. And, in fact, it has been determined that today's electronic digital computer functions most efficiently using a Binary system for its internal functions. This base-two system follows naturally from the fact that the computer circuitry is made up, in large part, of bi-stable devices.

The Binary System

The Binary System is, of course, a system which utilizes only two numeric characters, 0 and 1. The principles of counting and arithmetic are exactly the same as for decimal numbers. Counting proceeds in the order, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, etc. Whereas, in the decimal system every tenth count in a digit position causes a change in the next position to its left; in the Binary system every second count steps the next higher digit position. Each successive position to the left represents the next higher power of two. Thus, the Binary number, 10110, is equal to 1 times 2^4 plus 0 times 2^3 plus 1 times 2^2 plus 1 times 2^1 plus 0 times 2^0. This is equivalent to the decimal number 22.

Similarly, in a fractional Binary number, the positions to the right of the Binary point represent successive negative powers of two. Thus, the Binary fraction, .10110, represents 1 times 2^{-1} plus 0 times 2^{-2} plus 1 times 2^{-3} plus 1 times 2^{-4} plus 0 times 2^{-5}. This is equivalent to the decimal, 1/2 plus 1/8 plus 1/16, or 11/16.

If we were to perform all our manual computations in Binary arithmetic, we would find that, although the individual operations are childishly simple, we would soon be drowning in a sea of 0's and 1's. The decimal number, 999,999, if represented in Binary, requires 20 digits. The computer, on the other hand, finds it much easier to handle a large number of digit positions than to deal with more than two digit values. In order to facilitate communication between the programmer and the computer, it is desirable to use a number system which provides for brief numerical notation; and one which permits easy conversion to and from the Binary system.

The Hexadecimal System

If we look carefully at the binary system, we note that four digit positions are sufficient to hold 16 numerical values from 0 to 15. The RPC-4000 has a word length of 32 bits (binary digits), so that one word can be divided evenly into eight 4-bit groups. If we select, for communication with the computer, a number system to a base of 16, we can represent a full computer word with just eight characters, and each character will correspond exactly with a unique pattern of binary digits.

This base-sixteen system is known as a Hexadecimal system. It is the system used for all direct communication with the computer. The table of Hexadecimal characters and their decimal and binary equivalents is as follows:

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Binary Coded Decimal

Frequently encountered in discussions of programming and computers is another, and somewhat different, system of numbers known as Binary Coded Decimal. This is actually a system built on two bases, 2 and 10. Base 2 is used in the Binary representation of each of the ten numeric characters, 0 thru 9. Normally, the Binary numbers, 0000 thru 1001 are used, and each 4-bit group signifies one Decimal character. The positional notation used is the same as for Decimal numbers in that each successive 4-bit Binary group proceeding to the left of the units position represents the next higher power of ten. Thus the Decimal number, 369, appears in Binary Coded Decimal as 0011 0110 1001. The same number in Binary is written 101110001.

The usefulness of this number system is in providing a convenient form in which to process numerical data through the Input/Output equipment. Its use, however, requires Input and Output program routines which will make the conversion to and from a true Binary representation.

Conversion Between Systems

In working with programs and computers, it frequently becomes necessary to perform a manual conversion from one number system to another. This is particularly true in debugging, where a word displayed on the scope face must often be verified against a symbolically written program word.

Conversion of a Decimal integer into its Binary or Hexadecimal equivalent may be accomplished by a system of successive divisions of the Decimal integer by the base of the system into which we are converting. For example, consider the Decimal number, 229. If we divide by the Hexadecimal base, 16, we get 14, with a remainder of 5. The 5 represents the least significant digit of the Hexadecimal number we are computing. If we now divide the integral quotient, 14, by 16, we get 0, with a remainder of 14. The 14, expressed hexadecimally by the character, E, represents the most significant digit of the desired number, so that the Hexadecimal equivalent of 229 is E5. To avoid confusion when working with number conversions, the base of a number is usually represented by a subscript, so that the above numbers would be more clearly expressed as 229_{10} and $E5_{16}$. Let's consider another example, this time showing the conversion in tabular form. The Decimal integer 1386 would be converted as follows:

DECIMAL NUMBER	÷ 16 =	QUOTIENT	WITH	REMAINDER
1386_{10}		86_{10}		(LEAST SIGNIGICANT) A_{16}
86_{10}		5_{10}		6_{16}
5_{10}		0		(MOST SIGNIFICANT) 5_{16}

The Hexadecimal equivalent of 1386_{10} is $56A_{16}$. Note that the "A" in the above example is used as a Hexadecimal numeral, and is the equivalent of a Decimal 10.

Decimal integers may be converted to Binary in exactly the same fashion. Conversion of 52_{10} to Binary proceeds as follows:

DECIMAL NUMBER	÷ 2 =	QUOTIENT	WITH	REMAINDER
52_{10}		26_{10}		(LEAST SIGNIFICANT) 0_2
26_{10}		13_{10}		0_2
13_{10}		6_{10}		1_2
6_{10}		3_{10}		0_2
3_{10}		1_{10}		1_2
1_{10}		0_{10}		(MOST SIGNIFICANT) 1_2

The Binary equivalent of 52_{10} is shown to be 110100_2. With a little practice, a Binary number can be readily converted to Decimal by inspection or by referring to a table of powers of two. Converting the above Binary

VALUE TO DECIMAL IS JUST A MATTER OF ADDING 2^5 (32), 2^4 (16), AND 2^2 (4) TO ARRIVE AT THE VALUE, 52_{10}. FRACTIONAL VALUES MAY BE SIMILARLY HANDLED BY USING NEGATIVE POWERS OF TWO.

CONVERSION OF A DECIMAL FRACTION INTO HEXADECIMAL OR BINARY MAY BE ACCOMPLISHED BY A SYSTEM OF SUCCESSIVE MULTIPLICATIONS BY THE BASE INTO WHICH WE ARE CONVERTING. FOR EXAMPLE, CONSIDER THE FOLLOWING CONVERSION OF $.912_{10}$ INTO HEXADECIMAL:

DECIMAL NUMBER X 16 =	PRODUCT	WITH	INTEGRAL CARRY	=	HEXADECIMAL
$.912_{10}$	14.592_{10}		14_{10}		E_{16}
$.592_{10}$	9.472_{10}		9_{10}		9_{16}
$.472_{10}$	7.552_{10}		7_{10}		7_{16}
ETC.					

IN THIS PROCESS, THE FIRST INTEGRAL CARRY REPRESENTS THE MOST SIGNIFICANT DIGIT OF THE HEXADECIMAL FRACTION, AND THE FRACTION, $.912_{10}$, CONVERTS TO $.E97_{16}$, COMPUTED TO 3 PLACES.

SCALING

THE NUMBERS, $.752_{10}$, 7.52_{10}, 75.2_{10}, AND 752_{10} ARE IDENTICAL IN APPEARANCE EXCEPT FOR THE PLACEMENT OF AN INSIGNIFICANT LOOKING SYMBOL KNOWN AS A DECIMAL POINT. YET THE USE OF THIS SYMBOL CAN MAKE A HUGE DIFFERENCE IN THE MEANING OF THE NUMBER IN WHICH IT IS PLACED. WHEN WE WORK WITH NUMBERS IN THIS FASHION, WE ARBITRARILY PLACE THIS POINT IN SUCH A WAY AS TO ESTABLISH A DESIRED MAGNITUDE FOR THE NUMBER WE WISH TO REPRESENT. THE DIGITS TO THE LEFT OF THIS POINT REPRESENT INTEGRAL VALUES, AND THE DIGITS TO THE RIGHT REPRESENT FRACTIONAL VALUES. IN ORDER TO PERFORM VALID ARITHMETIC OPERATIONS WITH THESE NUMBERS, WE MUST NECESSARILY BE COGNIZANT OF THE LOCATION OF THE POINT IN ALL OUR OPERANDS.

THE RPC-4000 COMPUTER WORD IS 32 BINARY DIGITS IN LENGTH. FOR A DATA WORD, THESE BITS PROVIDE FOR A SIGN BIT IN POSITION 0, AND 31 MAGNITUDE BITS. THE PRINCIPLES OF SCALING, OUTLINED ABOVE WITH RESPECT TO DECIMAL ARITHMETIC, APPLY ALSO TO THE BINARY VALUES USED BY THE COMPUTER. THE BINARY POINT DOES NOT ACTUALLY EXIST WITHIN THE COMPUTER. IT EXISTS ONLY IN THE MIND OR ON THE PAPER OF THE PROGRAMMER. IT IS HIS RESPONSIBILITY TO CONTROL THE PLACEMENT OF VALUES IN THE COMPUTER WORD SO THAT THESE IMPLIED BINARY POINTS WILL FALL IN THE PROPER POSITIONS TO PRODUCE VALID ARITHMETIC RESULTS.

THE BINARY POINT LOCATION BETWEEN BITS 0 AND 1 OF THE COMPUTER WORD, REFERRED TO AS THE MACHINE POINT, SERVES AS A REFERENCE POINT IN SPECIFYING THE SCALE FACTOR FOR A VALUE IN THE WORD. THE SYMBOL, "Q", HAS BEEN ESTABLISHED BY CONVENTION TO DENOTE THE PLACEMENT OF THE IMPLIED BINARY POINT WITH RESPECT TO THE MACHINE POINT. IF A VALUE IS ENTERED INTO THE COMPUTER AT A "Q" OF 5, THE BITS IN POSITIONS 1 THRU 5 REPRESENT THE INTEGRAL PORTION OF THE VALUE, AND THE BITS IN POSITIONS 6 THRU 31 REPRESENT THE FRACTIONAL PORTION.

TO PERFORM A VALID ADDITION OR SUBTRACTION IN THE COMPUTER, THE BINARY POINTS MUST BE LINED UP THE SAME AS WE WOULD LINE UP THE POINTS IN PERFORMING THESE OPERATIONS WITH PENCIL AND PAPER. THAT IS, THE OPERANDS MUST BE STORED IN THE COMPUTER AT THE SAME "Q". A NUMBER AT A "Q" OF 12, ADDED TO ANOTHER NUMBER AT A "Q" OF 12, WILL PRODUCE A SUM WHICH IS ALSO AT A "Q" OF 12.

IN MULTIPLICATION HOWEVER, THE POINTS NEED NOT BE LINED UP, BUT MAY BE AT ANY LOCATION WHICH WILL PRODUCE A PRODUCT AT THE REQUIRED "Q". THE "Q" OF THE PRODUCT OF A MULTIPLICATION IS THE SUM OF THE "Q'S" OF THE MULTIPLICAND AND THE MULTIPLIER. A NUMBER AT A "Q" OF 6, MULTIPLIED BY A NUMBER AT A "Q" OF 3, WILL PRODUCE A PRODUCT AT A "Q" OF 9.

THE RULES FOR PERFORMING A DIVISION IN THE COMPUTER ARE EQUALLY SIMPLE, BUT REQUIRE ONE ADDITIONAL PRECAUTION ON THE PART OF THE PROGRAMMER. THE "Q" OF THE QUOTIENT IS DETERMINED BY SUBTRACTING THE "Q" OF THE DIVISOR FROM THE "Q" OF THE DIVIDEND. THE VALUE, 6, AT A "Q" OF 3, DIVIDED BY THE VALUE, 2, AT A "Q" OF 2, SHOULD PRODUCE A QUOTIENT, 3, AT A "Q" OF 1. BUT THE VALUE, 3, CAN NOT BE HELD AT A "Q" OF 1, SO THAT A CONDITION KNOWN AS A "DIVIDE CHECK" OCCURS. THIS, OF COURSE, PRODUCES AN OVERFLOW OUT OF THE ACCUMULATOR WHICH WILL TURN ON AN INDICATOR, KNOWN AS THE BRANCH CONTROL. TO AVOID THIS SITUATION, THE PROGRAMMER MUST SCALE HIS OPERANDS SO THAT THE QUOTIENT WILL FIT IN THE ACCUMULATOR. IN THE ABOVE EXAMPLE, SHIFTING THE DIVIDEND, 6, TO A "Q" OF 4, AND THEN DIVIDING BY 2 AT A "Q" OF 2, WILL PRODUCE A QUOTIENT OF 3, AT A "Q" OF 2, WHICH CAN BE CONTAINED WITHOUT OVERFLOW. CONSIDERED FROM THE STANDPOINT OF THE COMPUTER WORD, ITSELF, AND DISREGARDING THE IMPLIED BINARY POINT LOCATION, THE DIVISOR MUST ALWAYS EXCEED THE DIVIDEND TO PERFORM A VALID DIVIDE OPERATION.

THE MEANS FOR CONTROLLING THE SCALING OF VALUES IN THE COMPUTER ARE PROVIDED BY A SHIFT INSTRUCTION, WHICH PERMITS THE PROGRAMMER TO ADJUST THE "Q" OF ANY VALUE BY SHIFTING THE ACCUMULATOR CONTENTS RIGHT OR LEFT BY A PRESCRIBED NUMBER OF BIT POSITIONS. (SEE PAGE 38) IT IS IMPORTANT, IN CODING A PROGRAM, TO NOTE ON THE CODING SHEET THE "Q" VALUES IN ALL ARITHMETIC OPERATIONS.

3 THE CENTRAL COMPUTER

The basic unit of information in the computer is referred to as a Word. The computer Word in the RPC-4000 consists of 32 Binary Digits, commonly called "bits". This 32 bit pattern of information has a meaning dependent upon the context in which it is used. That is to say, that the same word from memory may be used as an instruction or as an arithmetic operand, and this usage is determined by the manner in which the program is written.

When used as numerical data, a Word is considered as consisting of a sign bit in the left-most position, followed by 31 magnitude bits.

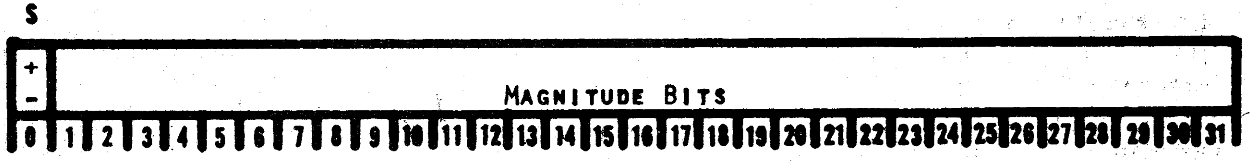


Figure 2. RPC-4000 Data Word Format

When used as an instruction, a Word is considered as consisting of a Command in bits 0 thru 4, an Operand or an Operand Address in bits 5 thru 17, the Next Instruction Address in bits 18 thru 30, and the Index Tag in bit 31.

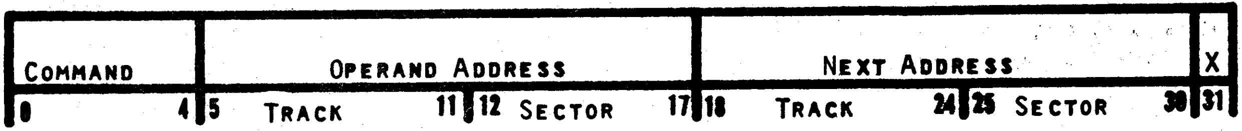
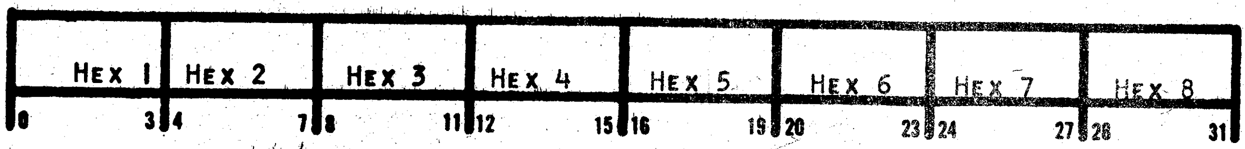


Figure 3. RPC-4000 Instruction Word Format

In certain cases it is convenient to consider the 32 bits of a Word as eight Hexadecimal characters. This is particularly true in entering information manually into the computer or in analyzing certain computer outputs.



Hex 1	Hex 2	Hex 3	Hex 4	Hex 5	Hex 6	Hex 7	Hex 8
0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31

Figure 4. RPC-4000 Hexadecimal Word Format

THE MEMORY DRUM

Information is stored in the RPC-4000 on a magnetic drum which rotates at the rate of 3600 revolutions per minute. This information will consist of both data words and instruction words. Each individual bit of each word stored in the computer is in the form of a discrete magnetized spot on the iron oxide coated drum surface. Each bit can exist in one of two magnetic states representing the binary values, 0 (zero) and 1 (one). All information in the computer, including all memory registers and all working registers, is represented in this manner.

The computer words are arranged in parallel bands around the drum, each band containing 64 word positions. Associated with each of these bands is a magnetic read/write head (those bands used for double access each have two read/write heads). The band of information traced out by any given head is referred to as a Track. Each of the 64 word positions around the circumference of the drum is referred to as a Sector.

The RPC-4000 memory consists of 128 tracks (numbered 000 thru 127) and 64 sectors (numbered 00 thru 63). Hence, the location of any word in memory can be specified by its track and sector number. This number is known as the Address of the word in question.

Tracks 000 thru 122 of the magnetic drum are the Main Memory storage area for the RPC-4000. Each track has 64 associated sectors, each of which contains one computer word. The total storage capacity of Main Memory is 7872 words. Any word in Main Memory may be referenced by specifying its track and sector position. Thus, 09843 refers to the word whose address is Track 098, Sector 43.

Maximum access time for a word in Main Memory is the time required for one complete drum revolution, or approximately 17 milliseconds. Average access time is approximately 8.5 milliseconds.

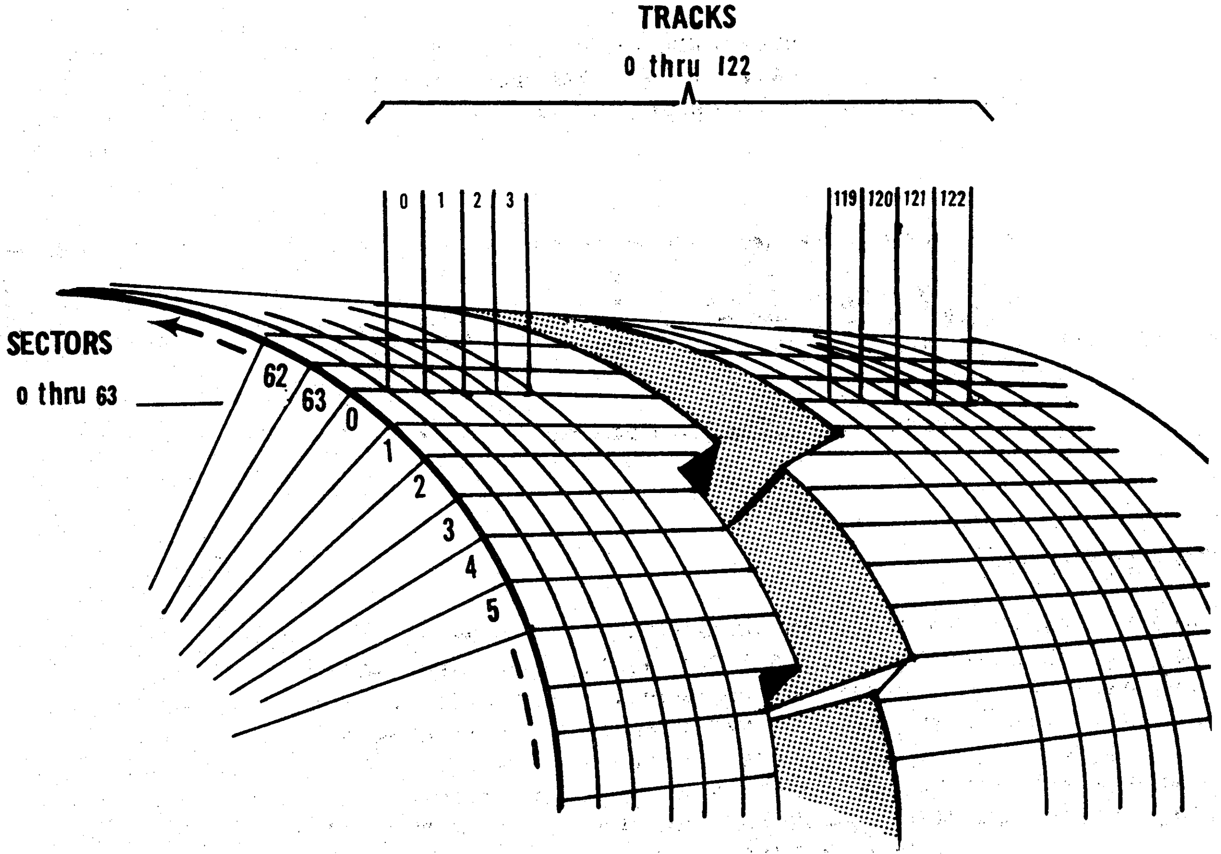


FIGURE 5. RPC-4000 MAIN MEMORY STORAGE

DUAL ACCESS TRACKS

IN ADDITION TO THE MAIN MEMORY JUST DISCUSSED, THERE ARE TWO BANDS OF MEMORY STORAGE ON THE DRUM FOR WHICH DUAL ACCESS IS PROVIDED. EACH OF THESE BANDS HAS TWO READ/WRITE HEADS, ADDRESSED BY THEIR OWN INDIVIDUAL TRACK NUMBERS. THE HEADS ADDRESSED AS TRACK 123 AND TRACK 125 SHARE THE SAME STORAGE BAND. THE TRACK 125 HEAD IS DISPLACED SO AS TO BE 16 WORD TIMES LATER THAN THE TRACK 123 HEAD. THEREFORE, BOTH HEADS REFER TO THE SAME SET OF WORDS, AND THE WORD WHICH IS ADDRESSED BY 12301 IS THE SAME WORD AS THAT ADDRESSED BY 12517.

LIKEWISE, THE HEADS ADDRESSED AS TRACK 124 AND TRACK 126 SHARE THE SAME SET OF WORD POSITIONS. IN THIS CASE, THE TRACK 126 HEAD IS DISPLACED SO AS TO BE 24 WORD TIMES LATER THAN THE TRACK 124 HEAD. THE WORD ADDRESSED BY 12401 IS THE SAME WORD AS THAT ADDRESSED BY 12625.

Random access time for a word from the Dual Access storage area is the same as for a word from Main Memory. Total Dual Access storage capacity is 128 words.

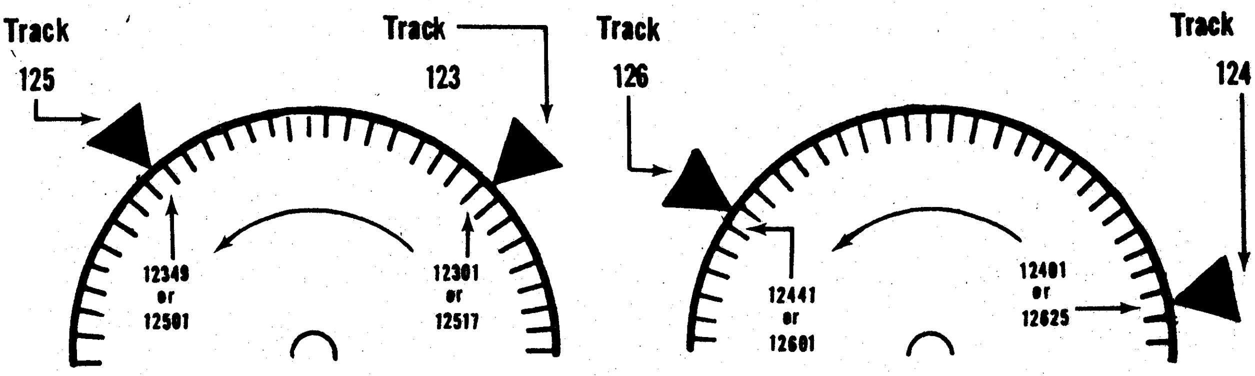


Figure 6. Dual Access Storage Tracks

The Recirculating Line

There are 8 additional words of memory storage on the drum in what is known as the Recirculating Line. The associated read/write head is referred to as Track 127, and this track provides a special fast access storage area. The fast access capability derives from the fact that these 8 words are repeated 8 times around the drum. This is accomplished in the following manner.

In addition to, and flanking the normal read/write head are a separate read head and write head. The read head is placed so as to trail the write head by 8 words. That is to say, that a word position on the drum will reach the write head 8 words before reaching the read head. Consequently, as each word is read by the read head, it is immediately written back on the drum 8 word positions farther along on the drum. Likewise, any new word value written on the drum by the normal read/write head will be picked up when it reaches the read head and recirculated.

Since every eighth word around the drum on Track 127 is identical, sector addresses will be modulo 8. The addresses 12701, 12709 and 12717 will all refer to the same word inasmuch as the sector addresses differ by multiples of 8. With 8 accesses provided for each word every drum revolution, maximum access time for the recirculating line is approximately 2 milliseconds, with an average access time of about 1 millisecond.

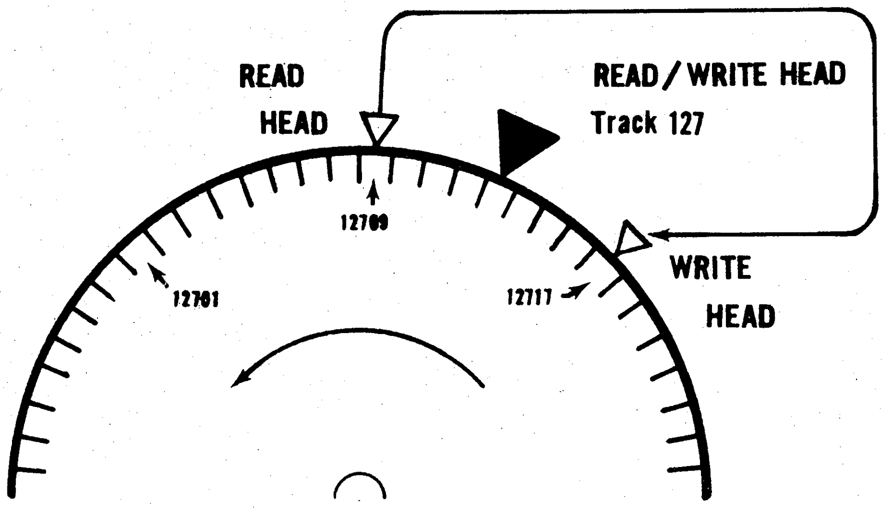


Figure 7. The Recirculating Line

The Sector Reference Timing Track

Although this track cannot be read directly or modified by the programmer, some explanation of its function may be helpful. The Sector Reference Timing Track contains the Sector Reference numbers 00 thru 63, which are permanently pre-recorded at the time of manufacture. The track has a read head only, which serves to locate, by reference to this Timing Track, any Sector Address on the drum which is called for in a program.

During a Compare instruction (CME or CMG) the contents of this track are continually copied into bits 25 thru 30 of the Index Register until a successful comparison is found, or until completion of the instruction, at which points any further copying is inhibited. Consequently, following a successful comparison, the Sector Address of the memory word which compares can be determined by subtracting 1 from the Timing Track value in the Index Register.

THE WORKING REGISTERS

A Register may be described as the hardware for storing a single computer word. Those registers which are used to perform the various arithmetic and control functions within the computer are the Working Registers. There are four such registers in the RPC-4000, each in the form of a recirculating line on the drum with a one word spacing.

These four Working Registers together comprise what is referred to as the Computing Control Unit. As such, they execute all internal data processing, AND CONTROL INPUTS AND OUTPUTS TO AND FROM THE COMPUTER. THE REGISTERS ARE KNOWN INDIVIDUALLY AS:

U-----UPPER ACCUMULATOR
L-----LOWER ACCUMULATOR
C-----COMMAND REGISTER
X-----INDEX REGISTER

THE UPPER ACCUMULATOR

GENERALLY SPEAKING, THE UPPER ACCUMULATOR IS THAT REGISTER IN THE COMPUTING CONTROL UNIT WHICH HOLDS THE COMPUTER WORD TO BE OPERATED UPON. AT THE COMPLETION OF AN ARITHMETIC OR LOGICAL OPERATION, IT WILL NORMALLY HOLD THE RESULT OF THE OPERATION.

THE UPPER ACCUMULATOR IS THE PRIMARY WORKING REGISTER. IT CAN RECEIVE INFORMATION FROM, OR SEND INFORMATION TO ANY REGISTER IN MEMORY, AND ANY WORKING REGISTER EXCEPT THE COMMAND REGISTER. NORMALLY THESE ARE FULL WORD TRANSFERS. HOWEVER, INFORMATION MAY BE SHIFTED FROM THE UPPER TO THE LOWER ACCUMULATOR, OR FROM THE LOWER TO THE UPPER ACCUMULATOR ON A BIT-BY-BIT BASIS.

FOR ANY ARITHMETIC OPERATION, THE CONTENTS OF THE UPPER ACCUMULATOR ARE CONSIDERED AS CONSISTING OF A SIGN BIT FOLLOWED BY 31 MAGNITUDE BITS AND THE OPERATION WILL BE PERFORMED ACCORDING TO THE NORMAL LAW OF SIGNS. FOLLOWING A MULTIPLY (MPY) INSTRUCTION, THE UPPER ACCUMULATOR WILL CONTAIN THE MOST SIGNIFICANT HALF OF THE DOUBLE WORD PRODUCT. BEFORE A DIVIDE (DIV) INSTRUCTION, THE UPPER ACCUMULATOR WILL CONTAIN THE MOST SIGNIFICANT HALF OF THE DOUBLE WORD DIVIDEND.

THE LOWER ACCUMULATOR

THE LOWER ACCUMULATOR MAY BE THOUGHT OF AS A SUPPLEMENT TO THE UPPER, PROVIDING AN ALTERNATE DATA HANDLING REGISTER, AND SERVING AS AN EXTENSION TO THE UPPER FOR BIT MANIPULATION AND EXTRA PRECISION ARITHMETIC. ITS CONTENTS MAY BE ADDED TO, SUBTRACTED FROM, AND TRANSFERRED DIRECT TO MEMORY, IN THE SAME WAY THAT THESE OPERATIONS ARE PERFORMED WITH THE UPPER.

BEFORE A DIVIDE (DIV) INSTRUCTION, THE LOWER ACCUMULATOR WILL CONTAIN THE LEAST SIGNIFICANT HALF OF THE DIVIDEND. FOLLOWING A DIVIDE (DIV) OR A DIVIDE UPPER (DVU) INSTRUCTION, THE LOWER ACCUMULATOR CONTAINS THE REMAINDER. FOLLOWING A MULTIPLY (MPY) INSTRUCTION, THE LOWER ACCUMULATOR CONTAINS THE LEAST SIGNIFICANT HALF OF THE PRODUCT.

WHEN COMBINED WITH THE UPPER ACCUMULATOR TO FORM A DOUBLE LENGTH WORD, BIT "0" OF THE LOWER ACCUMULATOR IS CONSIDERED AS A DATA BIT. OTHERWISE, IT IS CONSIDERED A SIGN BIT.

THROUGH THE USE OF A MODIFIER BIT IN THE EXCHANGE (EXC) INSTRUCTION, THE LOWER ACCUMULATOR CAN BE SET TO EIGHT WORDS INSTEAD OF ONE. THESE EIGHT

WORDS FUNCTION AS INDIVIDUAL ACCUMULATORS, BUT ARE SUBJECT TO CERTAIN RESTRICTIONS IN THEIR USE. FOR FURTHER DISCUSSION OF THIS MODE, SEE PAGE 21.

THE COMMAND REGISTER

THE COMMAND REGISTER IS THAT REGISTER WHICH HOLDS AN INSTRUCTION WORD DURING THE TIME THAT IT IS BEING INTERPRETED AND OPERATED BY THE COMPUTER. THE INSTRUCTION WORD CONSISTS OF FOUR LOGICAL PARTS OR FIELDS, EACH OF WHICH IS CONSIDERED WHEN AN INSTRUCTION IS PERFORMED:

1. THE COMMAND FIELD (BITS 0 THRU 4)---THIS FIELD IN THE COMMAND REGISTER WILL CONTAIN THE NUMERICAL CODE REPRESENTING THE OPERATION TO BE PERFORMED. IT SERVES ONLY TO IDENTIFY THE REQUIRED OPERATION AND HAS NO QUANTITATIVE SIGNIFICANCE.
2. THE DATA ADDRESS FIELD (BITS 5 THRU 17)---FOR MOST INSTRUCTIONS, THIS FIELD WILL CONTAIN A MEMORY ADDRESS SIGNIFYING THE LOCATION OF THE OPERAND. FOR SOME INSTRUCTIONS, THIS FIELD WILL CONTAIN A SET OF LOGICAL MODIFIERS WHICH SERVE TO FURTHER DEFINE THE OPERATION OF A BASIC COMMAND. AND FOR A FEW INSTRUCTIONS, THE <u>CONTENTS</u> OF THIS FIELD ARE USED AS THE OPERAND.
3. THE NEXT ADDRESS FIELD (BITS 18 THRU 30)---THIS FIELD WILL CONTAIN A MEMORY ADDRESS SIGNIFYING THE LOCATION OF THE INSTRUCTION WHICH IS TO OPERATE IMMEDIATELY FOLLOWING THE ONE CURRENTLY IN THE COMMAND REGISTER. THE ONLY EXCEPTION TO THIS IS THAT, UPON ENCOUNTERING AN ACTIVE TRANSFER INSTRUCTION, THE NEXT INSTRUCTION WILL BE TAKEN FROM THE ADDRESS SPECIFIED IN THE DATA ADDRESS FIELD.
4. THE INDEX TAG FIELD (BIT 31)---A "1" BIT IN THIS FIELD WILL CAUSE THE COMPUTER TO APPLY INDEXING TO THE INSTRUCTION WHEN IT IS PLACED IN THE COMMAND REGISTER. THIS MEANS THAT THE DATA ADDRESS FIELD IS INCREMENTED BY THE CONTENTS OF BITS 5 THRU 17 OF THE INDEX REGISTER. IF THE INDEX TAG IS SET TO "0", INDEXING IS NOT APPLIED, AND THE DATA ADDRESS FIELD IN THE COMMAND REGISTER WILL BE IDENTICAL WITH THE CORRESPONDING INSTRUCTION IN MEMORY.

THE INDEX REGISTER

THE INDEX REGISTER PERFORMS SEVERAL IMPORTANT FUNCTIONS IN THE RPC-4000. ITS PRIMARY USE IS FOR ADDRESS MODIFICATION AND, FOR THIS PURPOSE, BITS 5 THRU 17 OF THE INDEX REGISTER SERVE TO HOLD A VALUE BY WHICH THE DATA ADDRESS FIELD OF AN INSTRUCTION MAY BE INCREMENTED. THIS INCREMENTAL VALUE MAY BE PLACED IN THE INDEX REGISTER BY MEANS OF A LOAD INDEX (LDX) INSTRUCTION, AND CAN BE USED BY INCLUDING AN INDEX TAG IN THE APPROPRIATE INSTRUCTION.

BITS 18 THRU 24 OF THE INDEX REGISTER ARE USED TO HOLD THE REPEAT COUNT FOR ANY INSTRUCTION WHICH IS OPERATED IN THE REPEAT MODE. (SEE PAGE 22). THIS COUNT IS LOADED BY A LOAD COUNT (LDC) INSTRUCTION IMMEDIATELY PRECEDING THE INSTRUCTION TO BE REPEATED.

Bits 25 thru 30 of the Index Register are used in conjunction with the Compare Memory Equal (CME) and the Compare Memory Greater (CMG) instructions. At the beginning of a compare instruction, the Sector Reference Timing Track is copied into bits 25 thru 30 of the Index Register. If the instruction is executed in the repeat mode, this copying occurs prior to each iteration until a successful comparison is made or until the specified number of repeats is completed. If and when a successful comparison occurs, any further copying of the timing track is inhibited. The Index Register will then contain a value one sector greater than the sector location of the memory word which compared successfully.

In addition to the above functions, the full word Index Register may be used as a rapid access storage location and may be exchanged with the Upper Accumulator through the use of the Exchange (EXC) instruction.

The Branch Control

Although it is not a register, there is another information handling device within the computer which should be mentioned here. This is an internal toggle switch consisting of only one bit and known as the Branch Control toggle. It is automatically turned on when an overflow condition occurs. It may be set by the instructions, SNS, CXE, CME, and CMG. It may be sensed by the instruction, TBC (Transfer on Branch control).

PROGRAMMED OPERATING MODES

There are certain modifications which can be made under program control, which will alter the normal operating procedures for the Central Computer. They are concerned, for the most part, with the processing of multi-word blocks of information or tables of associated data. The first of these modifications provides the ability to change the form of the Lower Accumulator from one word to eight individual Lower Accumulators. The second enables automatic repetition of an instruction for a selected number of times using consecutive storage locations.

The Eight Word Lower Accumulator

The Lower Accumulator can be set to eight words or back to one word by means of two control bits which are a part of the Exchange (EXC) instruction word. The effect of setting the Lower to eight words is that, instead of the same value being recirculated into all 64 word positions, there are eight separate values considered, each of which is recirculated into eight word positions. In this respect, the eight word Lower is the same format as the recirculating memory line, Track 127.

When an instruction involving the eight word Lower is executed, the accumulator word location used will correspond to the sector address of the instruction's operand. To put it another way, if we let L_0, L_1, L_2,...L_7 designate the eight Lower Accumulators, then an instruction with a Data Address of 10300 would use or affect L_0, as would 10308, 10316, etc. Likewise, an instruction with a Data Address of 02703,would use or affect L_3, as would 05403, 06711, 11819, etc.

If overflow occurs in a Lower Accumulator word, it has no effect on the other seven words of the Lower, nor does it affect the Upper Accumulator. It does, however, turn on the Branch Control toggle.

The eight word Lower Accumulator can be used with the Input (INP) instruction to receive up to 64 4-bit characters or 42 6-bit characters. Inputs enter into the least significant end of L_0, and each succeeding character causes the preceding characters to be shifted left until all eight accumulators are filled. If the number of input characters exceeds the capacity of the eight word Lower, characters will be lost out of the most significant end of L_7.

The Automatic Repeat Mode

The execution phase of an instruction operated in this mode may be extended to any desired number of word times, not to exceed 128. A word time is that time required for one sector of the drum to pass beneath the read/write head. It is approximately .25 milliseconds. Thus, the number of repetitions of an instruction will be a function of the Repeat Count and the word times required to complete one execution. Most instructions require only one word time for command execution and, therefore, the number of repetitions beyond the initial execution will be equal to the specified Repeat Count.

The Repeat Mode is initiated by a Load Count (LDC) instruction, which places the Repeat Count in the Repeat Control field of the Index Register (bits 18 thru 24). The instruction immediately following the LDC instruction will be the only one affected. It will be placed in the Command Register and, where appropriate, the drum will be searched for its Data Address as in normal operation. But its execution phase will be continually repeated until the Repeat Count in the Index Register runs out. For each repetition, the Data Address used will be the one beneath the head at the time of execution. When this operation continues beyond Sector 63, it proceeds with Sector 00, 01, etc., of the same track. Track switching is not possible during a Repeat Mode operation.

The Repeat Count in the Index Register does not define the number of repetitions of an instruction whose execution phase requires more than one word time. The number of repetitions is determined by dividing the number of word times in the extended execution phase by the word times required for one execution. It should be understood that an improper Repeat Count set by the LDC instruction, can result in an incomplete execution of the repeated instruction.

Any instruction except LDC can be operated in the Repeat Mode. However, its use with some instructions leads to rather unpredictable results. Its primary use is with the arithmetic instructions for block processing or block transfers, and with the compare instructions, CME and CMG, for table search.

A special case occurs when a conditional transfer instruction, TBC or TMI, is operated in the repeat mode. If the transfer is inactive, no action is taken and the execution phase merely delays proceeding to the next instruction until the Repeat Count runs out. If the transfer is active, the normal execution phase is never reached. Consequently, the Repeat Count remains intact and causes the repetition of the instruction to which the transfer is made. This is the only instance in which the repeated instruction does not immediately follow the LDC instruction.

4 THE INSTRUCTION LIST

THE BASIC INSTRUCTIONS WHICH MAKE UP THE RPC-4000 INSTRUCTION LIST ARE DESCRIBED IN THE FOLLOWING PAGES. THESE ARE "MACHINE LANGUAGE" INSTRUCTIONS, EACH OF WHICH SPECIFIES A PARTICULAR ACTION TO BE TAKEN BY THE COMPUTER. PSEUDO-INSTRUCTIONS FOR COMMUNICATION WITH THE ROAR ASSEMBLY PROGRAM ARE NOT INCLUDED IN THIS SECTION.

A DRAWING OF THE INSTRUCTION WORD FORMAT ACCOMPANIES EACH DESCRIPTION. THIS DRAWING CONTAINS THE NUMERICAL COMMAND CODE AND ILLUSTRATES THE FIELD ALLOCATION FOR EACH COMPONENT OF THE INSTRUCTION. THE MNEMONIC DESIGNATOR FOR EACH COMMAND IS SHOWN IN BOLD TYPE AT THE TOP OF EACH DESCRIPTION.

IN MANY INSTANCES, DIAGRAMS HAVE BEEN INCLUDED TO SUPPORT AND CLARIFY THE TEXT EXPLANATIONS OF THE ACTIONS TAKEN BY THE INSTRUCTIONS. THESE DIAGRAMS INDICATE THE TRANSFER AND MODIFICATION OF THE CONTENTS OF THE VARIOUS REGISTERS WHICH ARE INVOLVED IN THE INSTRUCTION OPERATION.

THE "MINIMUM TIME" REFERRED TO IN THESE PAGES INCLUDES THE MINIMUM TIME TO COMPLETE ALL FOUR PHASES OF THE OPERATION, FROM THE BEGINNING OF THE SEARCH FOR THE INSTRUCTION WORD TO THE BEGINNING OF THE SEARCH FOR THE NEXT INSTRUCTION WORD.

HALT HLT

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
00	0 0 0	(Any)	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

THE COMPUTER WILL HALT WITH THE INSTRUCTION WORD IN THE COMMAND REGISTER.

IF INDEXING IS SPECIFIED, IT SHOULD BE NOTED THAT ANY INDEX VALUE WHICH RESULTS IN A NON-ZERO VALUE IN THE D-TRACK FIELD WILL EFFECTIVELY PRODUCE A SNS INSTRUCTION. (SEE NEXT PAGE.)

MINIMUM TIME---------4 WORD TIMES

OVERFLOW-------------NOT A FACTOR

BRANCH CONTROL-------NOT AFFECTED

REGISTERS AFFECTED---NONE

SENSE SNS

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
00	1 → 127	(Any)	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The initial action of the instruction is to turn off the Branch Control.

This instruction will sense the condition of certain manual switches as related to the D-track value. If one or more D-track bits correspond to a depressed Sense Switch, or if D-track $\geq 64_{10}$, the instruction will turn on the Branch Control.

Note that any index value which modifies the value of the D-track will alter the logical correspondence of the D-track bits and the Sense Switches.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Conditionally set "On"

Registers affected---None

COMPARE INDEX EQUAL **CXE**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
01	(Data Value)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The initial action of this instruction is to turn off the Branch Control. The Data Value in the instruction is then compared with bits 5 → 17 of the Index Register. If there is a one-to-one correspondence, the Branch Control is turned on. If not, the Branch Control remains off.

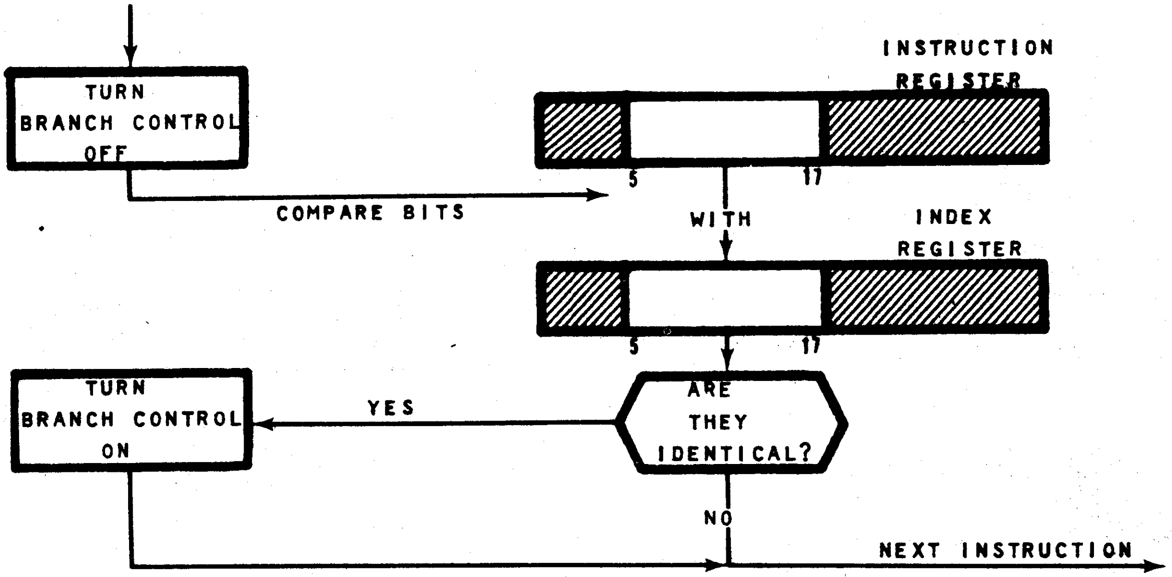
If indexing is specified in this instruction, a Data Value of zero will turn on the Branch Control, regardless of the index value. This occurs because the zero Data Value, when indexed, becomes identical with the Index Value. Conversely, any Data Value other than zero will turn off the Branch Control, regardless of the Index Value, inasmuch as any non-zero value, when indexed, becomes greater than the Index Value.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Conditionally set "On" or "Off"

Registers affected---None



RESET AND ADD TO UPPER **RAU**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
02	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The contents of the memory location specified by the Data Address will replace the current contents of the Upper Accumulator.

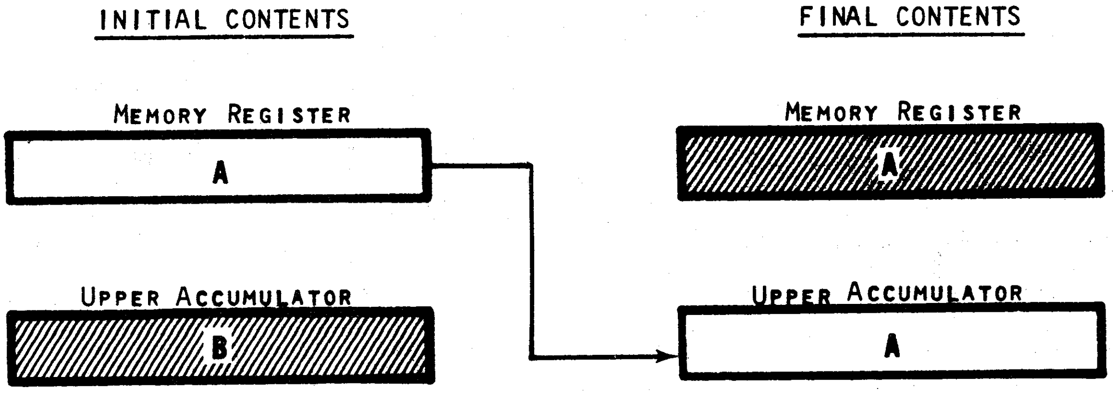
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Upper Accumulator



RESET AND ADD TO LOWER RAL

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
03	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The contents of the memory location specified by the Data Address will replace the current contents of the Lower Accumulator.

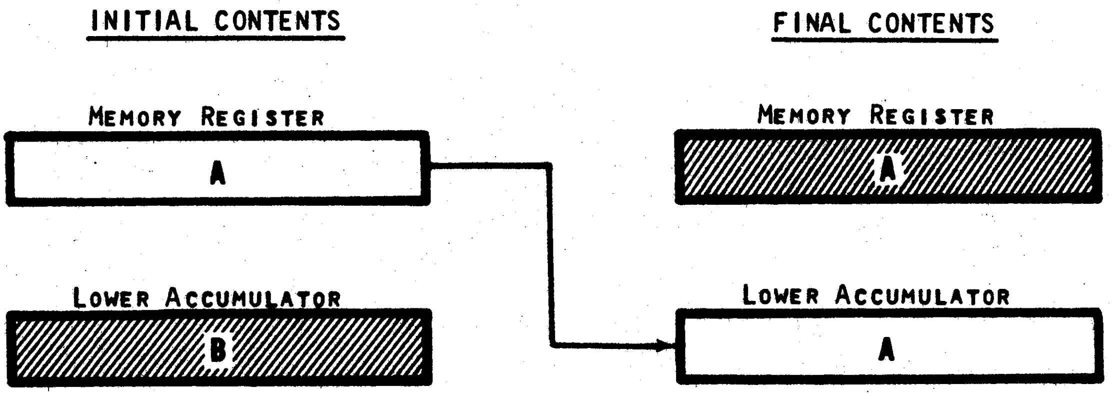
The Data Address may be modified by indexing.

Minimum time----------4 word times

Overflow--------------Not a factor

Branch Control--------Not affected

Registers affected---Lower Accumulator



STORE ADDRESS FROM UPPER **SAU**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
04	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores bits 5 → 17 of the Upper Accumulator into bits 5 → 17 of the memory location specified by the Data Address, replacing the current contents of these bits. The remainder of the memory word is left unchanged.

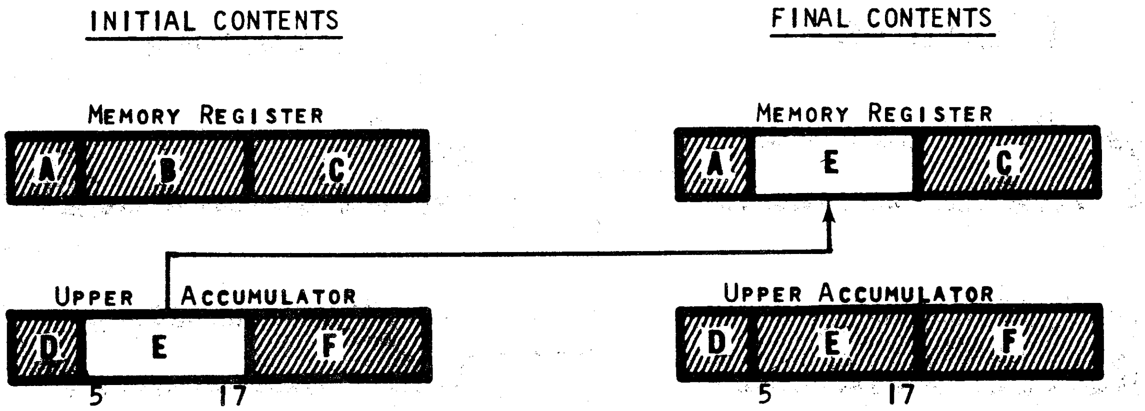
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control--------Not affected

Registers affected---Specified memory location



MASKED STORE MST

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
05	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores selected bits from the Lower Accumulator into the memory location specified by the Data Address, as masked by the Upper Accumulator. Where the upper Accumulator contains I's, stores the corresponding Lower Accumulator bits into the memory word. Where the Upper Accumulator contains 0's, leaves the corresponding memory word bits unaltered.

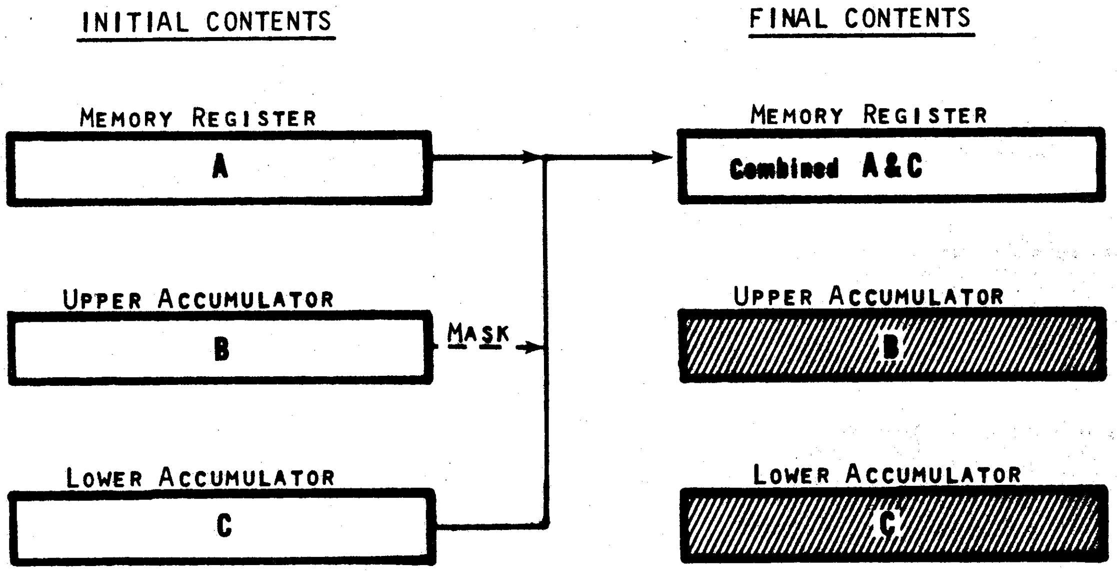
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Specified memory location



LOAD COUNT LDC

INSTRUCTION WORD FORMAT

Command	D Track	D-Sector	N-Track	N-Sector	X-Tag
06	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Loads bits 18 → 24 of the memory location specified by the Data Address into bits 18 → 24 of the Index Register, replacing the current contents of these bits. The remainder of the Index Register is left unchanged.

Causes the next instruction to be executed in the extended mode; that is, the instruction contained in the memory location specified by the Next Address will be repeated as many times as the number placed in bits 18 → 24 of the Index Register.

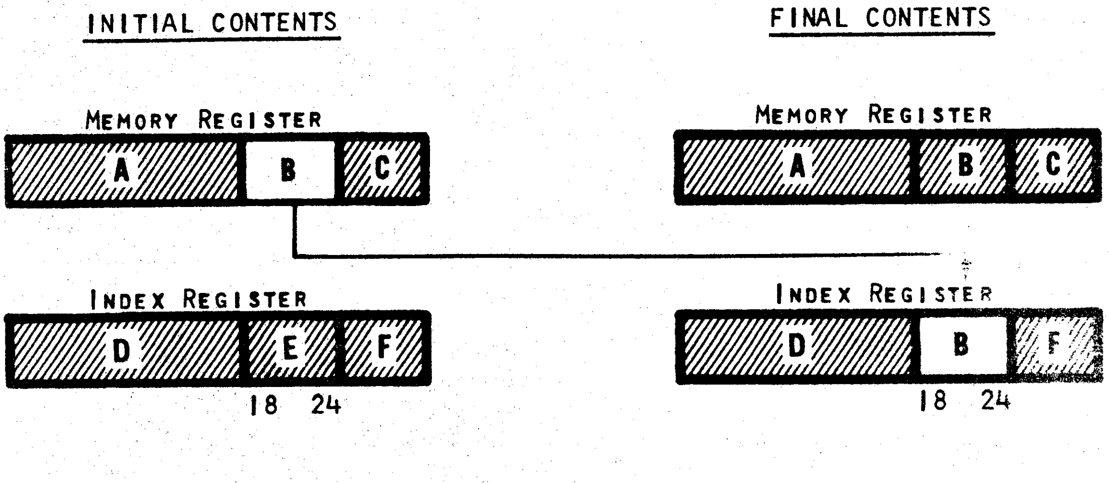
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not afffcted

Registers affected---Index Register



B = Number of times following instruction is repeated

LOAD INDEX LDX

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
07	(Data Value)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Loads the Data Value in the Instruction Word into bits 5→17 of the Index Register, replacing the current contents of these bits. The remainder of the Index Register is left unchanged.

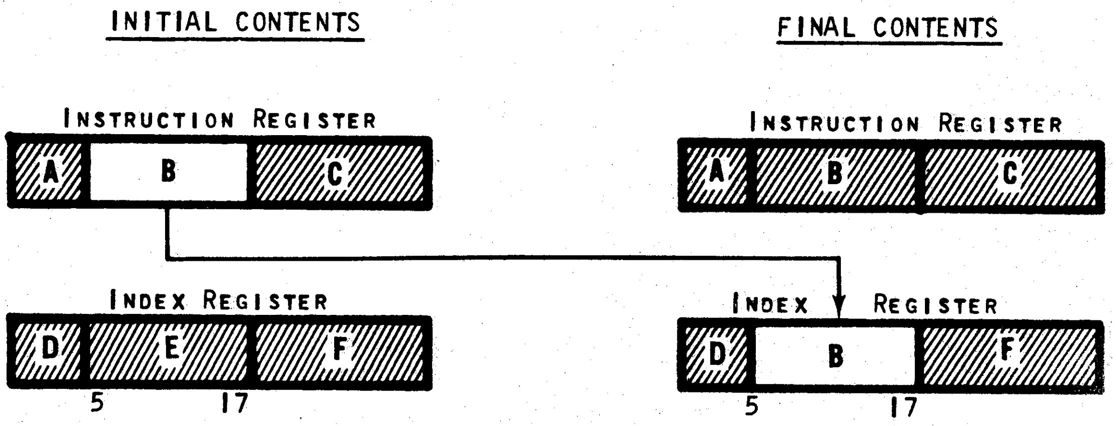
The Data Value may be modified by indexing. This will serve to increment the index value by the number in the Data Value field.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Index Register



INPUT

INP

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
08	000 or 064	▨	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the D-track of the Instruction Word contains 000, reads 4-bit characters into the accumulator. If the D-track contains 064, reads 6-bit characters into the accumulator. No other address is permitted in the D-track

If the Lower Accumulator is set at 1-word length, the characters will be read into the double length accumulator (combined Upper and Lower). If the Lower Accumulator is set at 8-word length, the characters will be read into the Lower Accumulator only.

Read-in begins with the low order character position of the low order accumulator word and, as subsequent characters are read in, the existing accumulator contents are shifted left one character position.

The D-track value may be modified by indexing. However, care should be taken that this does not result in a value other than 000 or 064.

Minimum time---------Dependent on the number of characters read

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Upper and/or Lower Accumulators

EXCHANGE

EXC

INSTRUCTION WORD FORMAT

Command	D-Track			D-Sector	N-Track	N-Sector	X-Tag
09	▨	0→2	0→15	(Any)	(Next Address)		0/1
0 4	5	6 7	8 11	12 17	18 24	25 30	31

This instruction performs one or more of the following functions, in accordance with the logical bit pattern in the D-track field.

If bit 6 = 1,---------Set the Lower Accumulator to 1-word length.

If bit 7 = 1,---------Set the Lower Accumulator to 8-word length.

(The above two functions are mutually exclusive.)

If bit 8 = 1,---------Replace the contents of the Upper Accumulator with the contents of the Index Register.

If bit 9 = 1,---------Replace the contents of the Index Register with the contents of the Upper Accumulator.

If bit 10 = 1,--------Replace the contents of the Upper Accumulator with the contents of the Lower Accumulator.

If bit 11 = 1,--------Replace the contents of the Lower Accumulator with the contents of the Upper Accumulator.

(Any or all of the above exchanges may be executed with the same instruction. If bits 8 and 10 are both set, the Upper Accumulator will receive the logical sum of the contents of the Index Register and the Lower Accumulator.)

The D-track and D-sector values may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Index Register and both Accumulators

DIVIDE UPPER

DVU

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
10	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

THE CONTENTS OF THE UPPER ACCUMULATOR ARE DIVIDED BY THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY THE DATA ADDRESS. THE QUOTIENT IS LEFT IN THE UPPER ACCUMULATOR, AND THE REMAINDER IS LEFT IN THE LOWER ACCUMULATOR. THE REMAINDER IS ALWAYS A POSITIVE VALUE, REGARDLESS OF THE SIGN OF THE QUOTIENT, SO THAT THE DIVIDEND MINUS THE REMAINDER WILL ALWAYS EQUAL THE DIVISOR TIMES THE QUOTIENT.

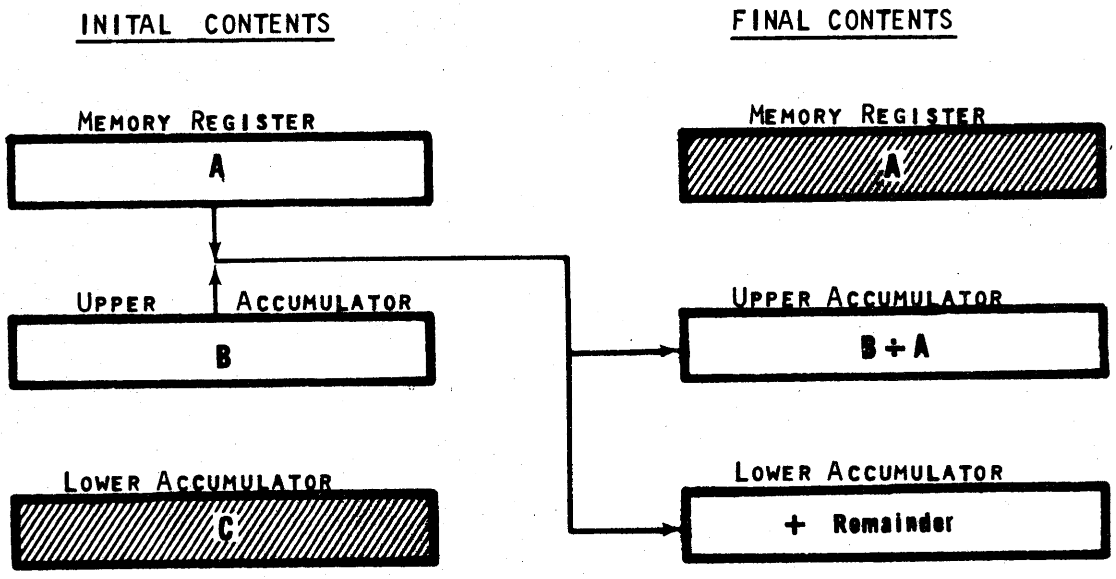
THE DATA ADDRESS MAY BE MODIFIED BY INDEXING.

MINIMUM TIME---------70 WORD TIMES

OVERFLOW-------------TURNS ON BRANCH CONTROL

BRANCH CONTROL-------TURNED ON BY OVERFLOW

REGISTERS AFFECTED---UPPER AND LOWER ACCUMULATOR



DIVIDE DIV

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
11	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

THE CONTENTS OF THE DOUBLE LENGTH ACCUMULATOR (COMBINED UPPER AND LOWER) ARE DIVIDED BY THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY THE DATA ADDRESS. THE QUOTIENT IS LEFT IN THE UPPER ACCUMULATOR, ANDTHE REMAINDER IS LEFT IN THE LOWER ACCUMULATOR. THE REMAINDER IS ALWAYS A POSITIVE VALUE, REGARDLESS OF THE SIGN OF THE QUOTIENT, SO THAT THE DIVIDEND MINUS THE REMAINDER WILL ALWAYS EQUAL THE DIVISOR TIMES THE QUOTIENT.

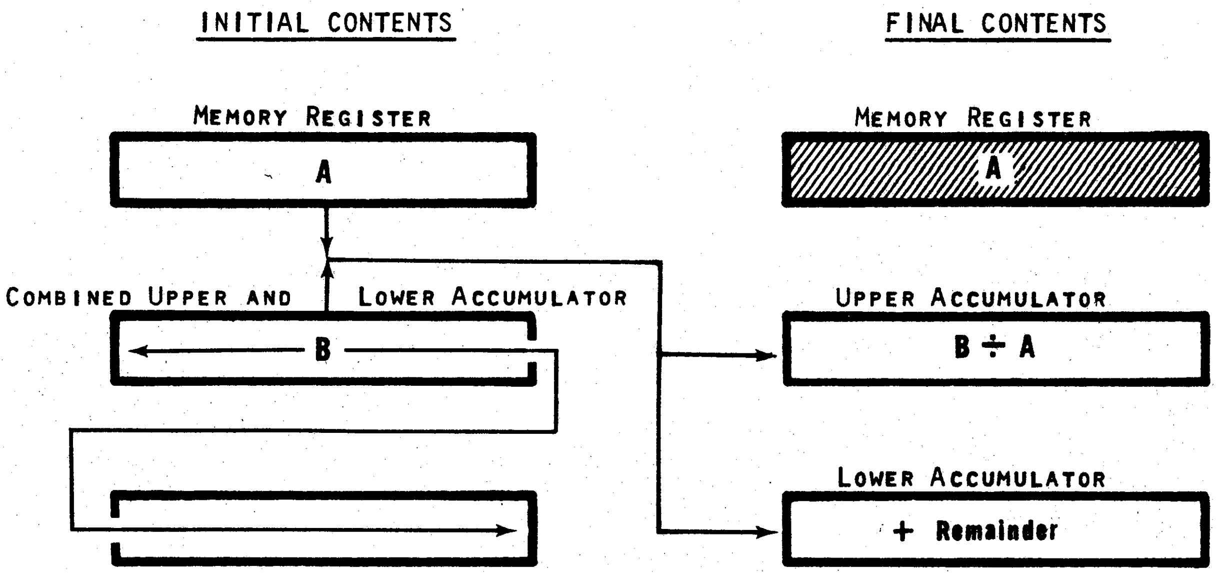
THE DATA ADDRESS MAY BE MODIFIED BY INDEXING.

MINIMUM TIME--------70 WORD TIMES

OVERFLOW------------TURNS ON BRANCH CONTROL

BRANCH CONTROL------TURNED ON BY OVERFLOW

REGISTERS AFFECTED--UPPER AND LOWER ACCUMULATOR



SHIFT RIGHT OR LEFT SRL

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
12	000 or 001	(Shift count)	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000, shifts the contents of the double length accumulator (combined Upper and Lower) to the right by the number of bit positions specified in the Shift Count. Bits shifted out of the low order bit position are lost.

If the D-track field contains 001, shifts the contents of the double length accumulator to the left by the number of bit positions specified in the Shift Count. If overflow occurs, turns on Branch Control. Bits shifted out of the sign position are lost.

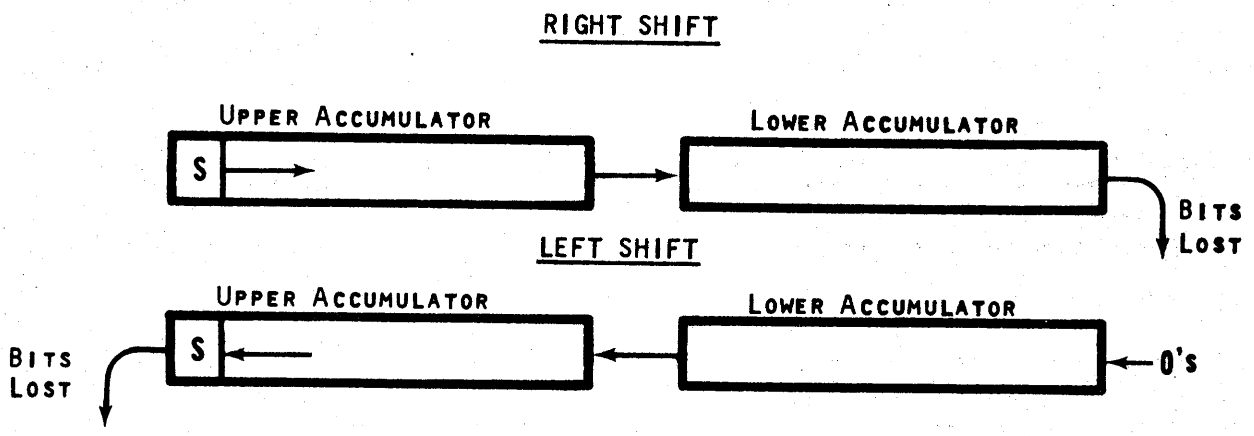
The Shift Count and/or direction of shift may be modified by indexing.

Minimum time---------7 word times plus 1 for each bit position shifted.

Overflow-------------Turns on Branch Control

Branch Control-------Turned on by overflow

Registers affected---Upper and Lower Accumulators



SHIFT LEFT AND COUNT SLC

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
13	▨	▨	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

THE VALUE CONTAINED IN THE DOUBLE LENGTH ACCUMULATOR (COMBINED UPPER AND LOWER) IS SHIFTED TO THE LEFT UNTIL IT IS IN NORMALIZED FORM; THAT IS, UNTIL BIT 1 CONTAINS THE FIRST SIGNIFICANT MAGNITUDE BIT. FOLLOWING THE SHIFT, THE LOWER ACCUMULATOR IS CLEARED TO ZERO, AND THE NUMBER OF BIT POSITIONS SHIFTED IS PLACED IN BITS 5→17 OF THE LOWER ACCUMULATOR. The D-sector address must be zero

INDEXING IS MEANINGLESS WITH THIS INSTRUCTION.

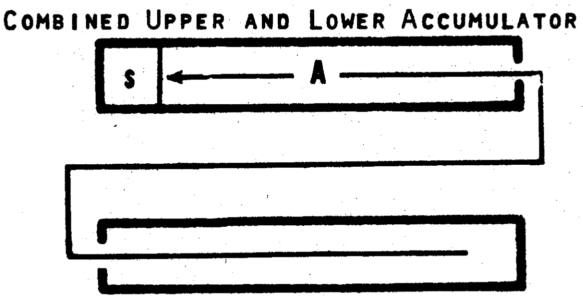
MINIMUM TIME---------7 WORD TIMES PLUS 1 FOR EACH BIT POSITION SHIFTED

OVERFLOW-------------NOT A FACTOR

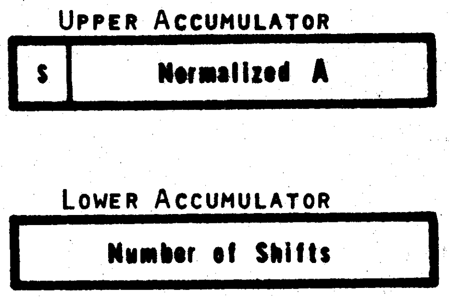
BRANCH CONTROL-------NOT AFFECTED

REGISTERS AFFECTED---UPPER AND LOWER ACCUMULATORS

INITIAL CONTENTS



FINAL CONTENTS



MULTIPLY MPY

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
14	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The contents of the Upper Accumulator are multiplied by the contents of the memory location specified by the Data Address. The resulting double length product is held in the combined Upper and Lower Accumulators.

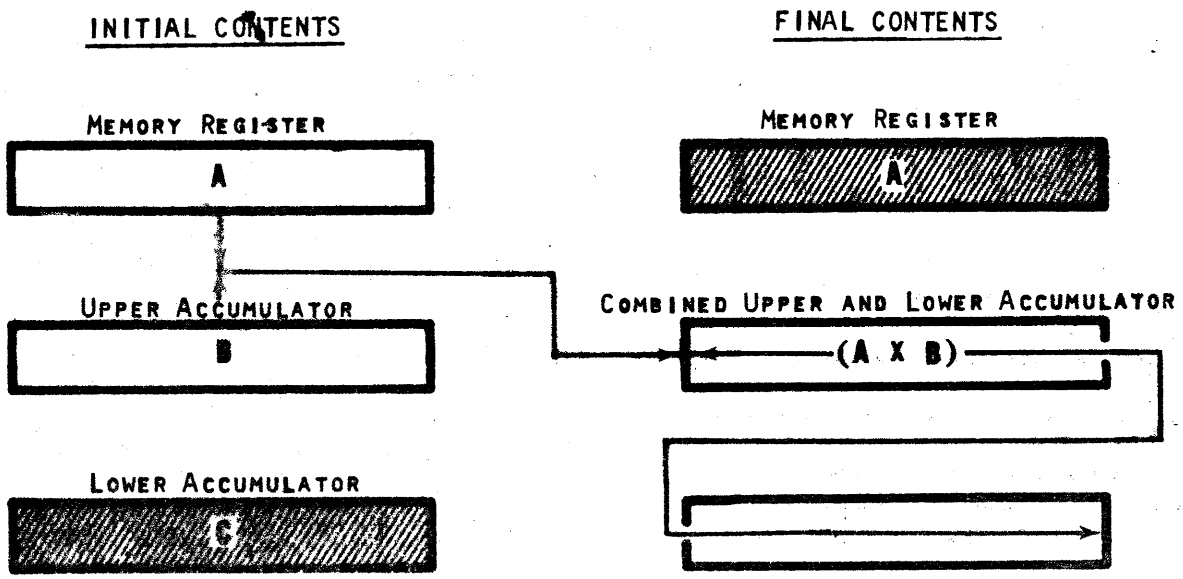
The Data Address may be modified by indexing.

Minimum time----------70 word times

Overflow-------------Not a factor

Branch Control--------Not affected

Registers affected---Upper and Lower Accumulators



MPT

MULTIPLY BY TEN

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
15	000 or 064	▨	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000, multiplies the contents of the Upper Accumulator by 10_{10}, retaining the product in the Upper Accumulator.

If the D-track field of the Instruction Word contains 064, multiplies the contents of the Lower Accumulator by 10_{10}, retaining the product in the Lower Accumulator.

The D-track value may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch control-------Not affected

Registers affected---Upper or Lower Accumulator

<u>PR</u>INT FROM <u>D</u>-ADDRESS PRD

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
16	000 → 127	(Any)	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000→063, prints the character it represents on the selected output device.

If the D-track field contains 064→127, selects the input and/or output devices it represents. (See selection table below)

The I/O interlock will be turned on, preventing the execution of another print instruction until completion of the current one, if the D-sector field contains any sector other than the first optimum sector (instruction location + 2). If this sector is specified, the interlock will be bypassed.

The D-track and D-sector values may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control--------Not affected

Registers affected---None

INPUT/OUTPUT SELECTION CODES

<u>D-track</u>	
64	Reader input
65	Reader input-----Punch output
66	Reader input-----Typewriter output
67	Reader input-----Punch and Typewriter output
68	Typewriter input
69	Typewriter input--Punch output
70	Typewriter input--Typewriter output
71	Typewriter input--Punch and Typewriter output
72	Photo-reader, Forward and Search

INPUT/OUTPUT SELECTION CODES (CONTINUED)

73	PHOTO-READER, REVERSE AND SEARCH
74	PHOTO-READER, FORWARD
75	PHOTO-READER, REVERSE
76-94	AVAILABLE FOR ADDITIONAL UNITS
95	MASTER RESET (RESET ALL UNITS)
96	AVAILABLE
97	PUNCH OUTPUT
98	TYPEWRITER OUTPUT
99	PUNCH AND TYPEWRITER OUTPUT
100	AVAILABLE
101	PUNCH OUTPUT
102	TYPEWRITER OUTPUT
103	PUNCH AND TYPEWRITER OUTPUT
104, 105	SEARCH MODE
106--124	AVAILABLE
125	COPY MODE ON
126	COPY MODE OFF
127	RESET OUTPUT UNITS

NOTES:

1. SELECTION OF A NEW INPUT DEVICE AUTOMATICALLY RESETS THE PREVIOUS ONE. ONLY ONE INPUT DEVICE MAY BE IN THE SYSTEM AT A TIME.

2. ANY COMBINATION OF OUTPUT DEVICES MAY BE INCLUDED IN THE SYSTEM AT ONE TIME. A RESET COMMAND IS NECESSARY TO DROP AN OUTPUT DEVICE FROM THE SYSTEM.

Print From Upper **PRU**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
17	000 → 127	(Any)	(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the D-track field of the Instruction Word contains 000 → 063; prints the character represented by the combination of bits 6 and 7 from the Instruction Word followed by the high order four bits of the Upper Accumulator.

If the D-track field contains 064 → 127, prints the character represented by the high order six bits of the Upper Accumulator.

The I/O interlock will be turned on, preventing the execution of another print instruction until completion of the current one, if the D-sector field contains any sector other than the first optimum sector (instruction location + 2). If this sector is specified, the interlock will be bypassed.

The D-track and D-sector values may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---None

EXTRACT **EXT**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
18	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Produces, in the Upper Accumulator, the logical product of the contents of the Upper Accumulator and the contents of the memory location specified by the Data Address. The resultant product will contain 1's in only those bit positions which are set to 1 in both the Upper Accumulator and the memory word.

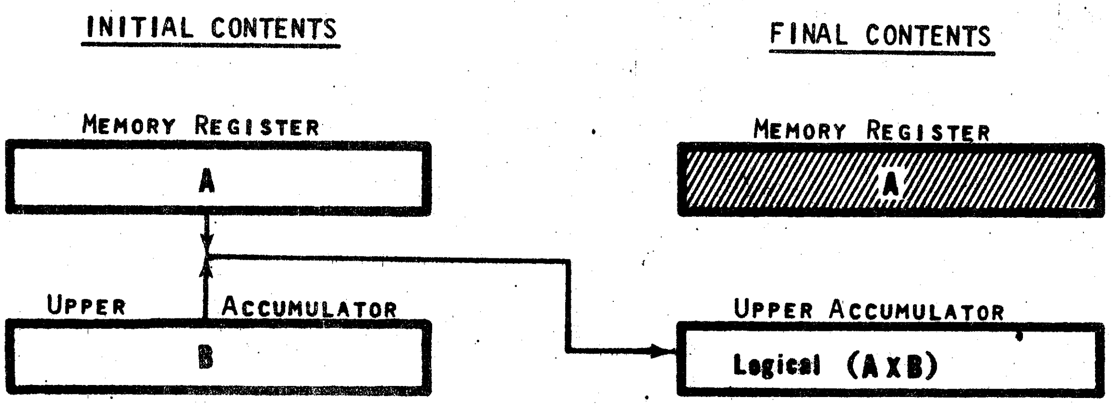
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control--------Not affected

Registers affected---Upper Accumulator



MASKED MERGE LOWER **MML**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
19	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

The contents of the memory location specified by the Data Address are merged with the contents of the Lower Accumulator under control of the Upper Accumulator. In those bit positions where the Upper Accumulator contains 0's, the Lower Accumulator is retained. In those bit positions where the Upper Accumulator contains 1's, the contents of the memory word replace the corresponding contents of the Lower Accumulator.

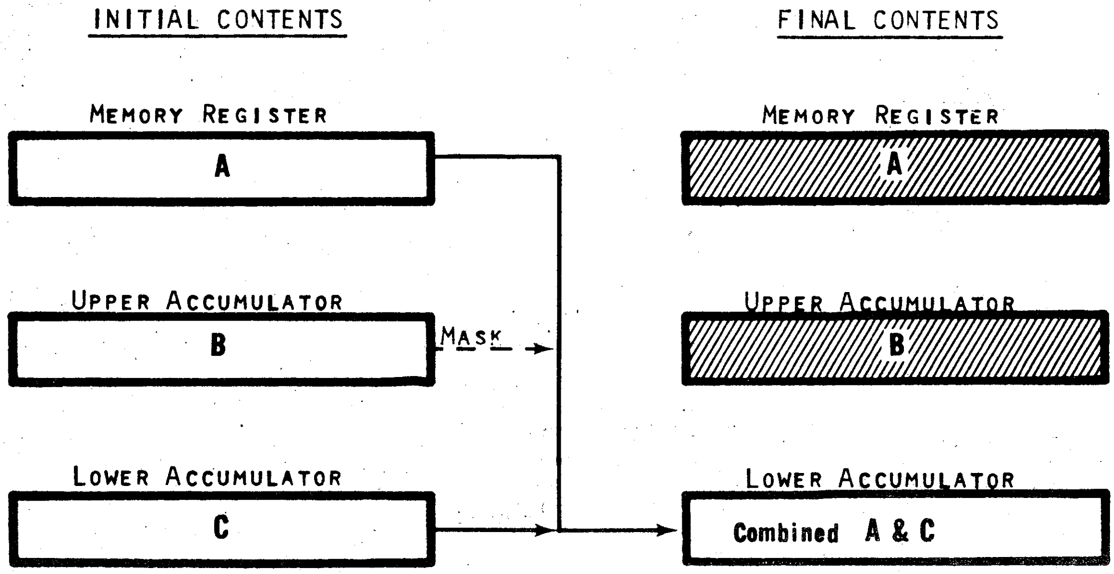
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---None



COMPARE MEMORY EQUAL CME

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
20	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Prior to performing the specified comparison, the Branch Control is turned off, and the Sector Reference Timing Track is copied into bits 25 ⟶ 30 of the Index Register. This will always be one sector ahead of the Data Address being compared.

Selected bits of the Upper Accumulator are then compared with corresponding bits in the memory location specified by the Data Address. Only those bit positions indicated by 1's in the Lower Accumulator mask are compared. If the selected values are identical, the Branch Control is turned on to indicate a successful comparison.

If the instruction is being executed in the Repeat Mode, the Sector Reference Timing Track is copied into bits 25 ⟶ 30 of the Index Register before each comparison until a successful comparison is made or until completion of the repeat function. A successful comparison inhibits any further copying of the timing track. The memory location containing the value which compares successfully may then be determined by reference to the Index Register.

The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Conditionally set "On" or "Off"

Registers affected---Index Register

Compare Memory Greater **CMG**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
21	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Prior to performing the specified comparison, the Branch Control is turned off, and the Sector Reference Timing Track is copied into bits 25 → 30 of the Index Register. This will always be one sector ahead of the Data Address being compared.

Selected bits of the Upper Accumulator are then compared with corresponding bits in the memory location specified by the Data Address. Only those bit positions indicated by 1's in the Lower Accumulator mask are compared. If the selected value in memory is greater than, or equal to the selected Upper Accumulator value, the Branch Control is turned on.

If the instruction is being executed in the Repeat Mode, the Sector Reference Timing Track is copied into bits 25 → 30 of the Index Register before each comparison until a memory value equal to, or greater than the accumulator value is found, or until completion of the repeat function. If such a value is found, any further copying of the timing track is inhibited. The memory location which contains this value can then be determined by reference to the Index Register.

The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Conditionally set "On" or "Off"

Registers affected---Index Register

TMI

TRANSFER ON MINUS

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
22	(Transfer Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the Upper Accumulator is negative (a "1" in bit position zero), control is transferred to the instruction specified by the Transfer Address. If the value is positive, the instruction has no effect, and the next instruction is that specified in the Next Address field.

The Transfer Address may be modified by indexing.

Minimum time----------4 word times

Overflow--------------Not a factor

Branch Control--------Not affected

Registers affected---None

TRANSFER ON BRANCH CONTROL TBC

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
23	(Transfer Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

If the Branch Control is on, control is transferred to the instruction specified by the Transfer Address, and the Branch Control is turned off. If the Branch Control is off, the instruction has no effect, and the next instruction is that specified in the Next Address field.

The Transfer Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Set to "Off"

Registers affected---None

STORE UPPER STU

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
24	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores the contents of the Upper Accumulator into the memory location specified by the Data Address, replacing its current contents.

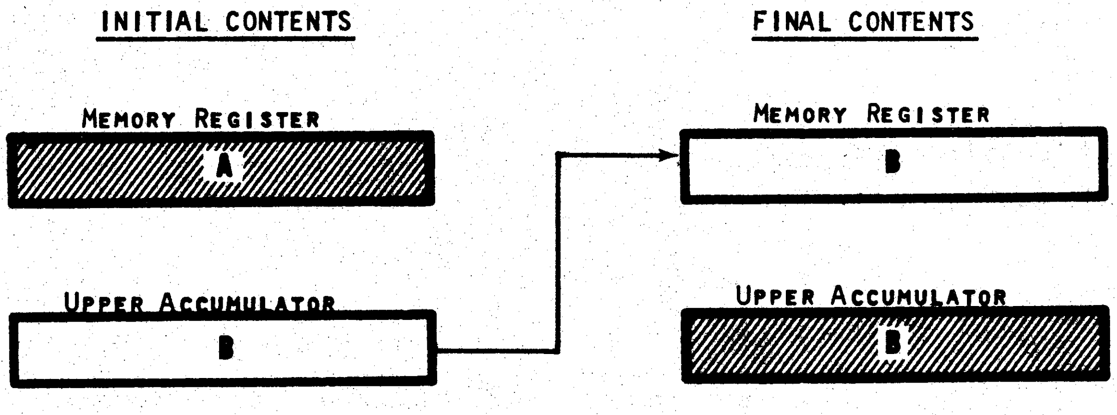
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Specified memory location



STORE LOWER STL

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
25	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores the contents of the Lower Accumulator into the memory location specified by the Data Address, replacing its current contents.

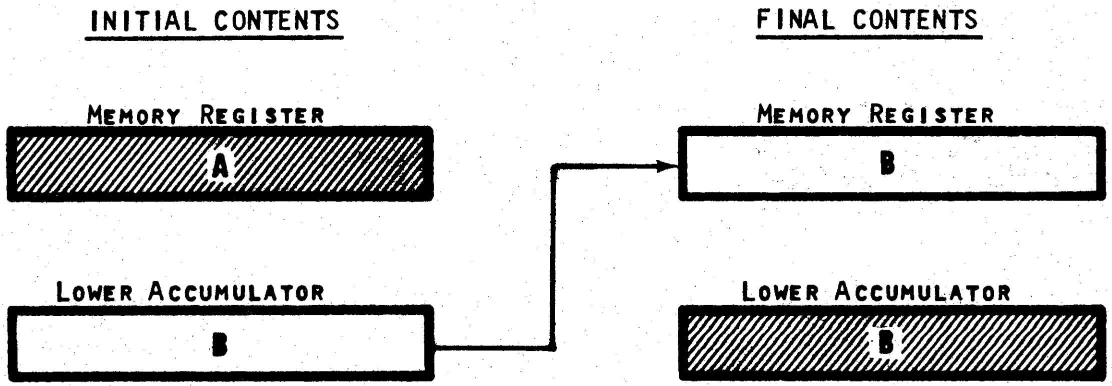
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Specified memory location



CLEAR UPPER CLU

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
26	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores the contents of the Upper Accumulator into the memory location specified by the Data Address, replacing the current contents. Clears the Upper Accumulator to zero.

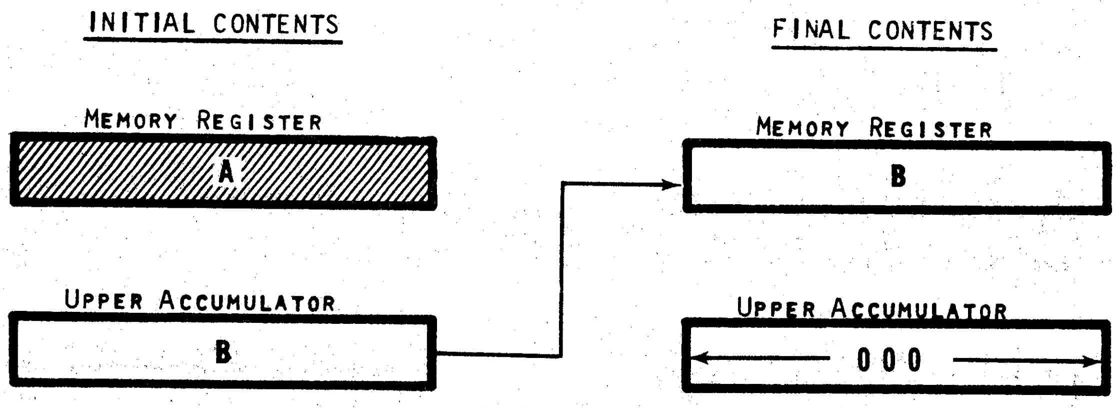
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Specified memory location and Upper
Accumulator



Clear Lower CLL

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
27	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Stores the contents of the Lower Accumulator into the memory location specified by the Data Address, replacing the current contents. Clears the Lower Accumulator to zero.

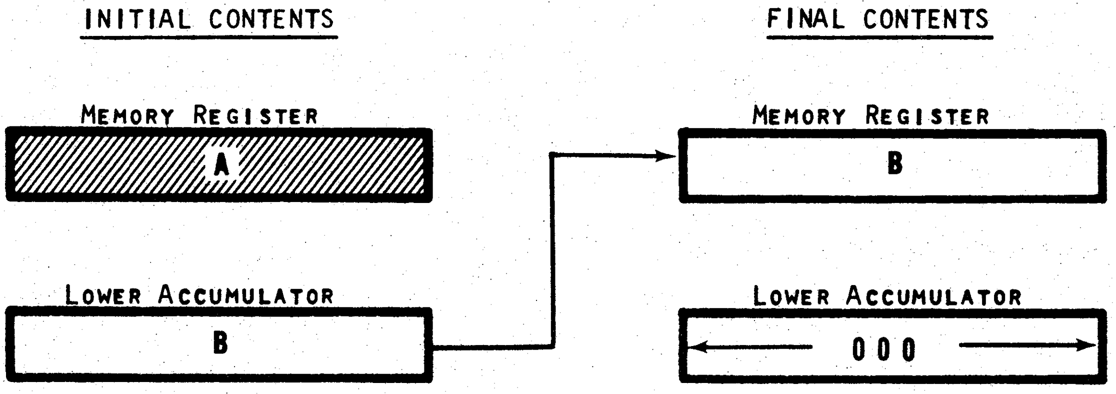
The Data Address may by modified by indexing.

Minimum time---------4 word times

Overflow-------------Not a factor

Branch Control-------Not affected

Registers affected---Specified memory location and Lower Accumulator



ADD TO UPPER ADU

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Ta
28	(Data Address)		(Next Address)		0/
0 4	5 11	12 17	18 24	25 30	

Adds the contents of the memory location specified by the Data Address to the contents of the Upper Accumulator. Retains the sum in the Upper Accumulator.

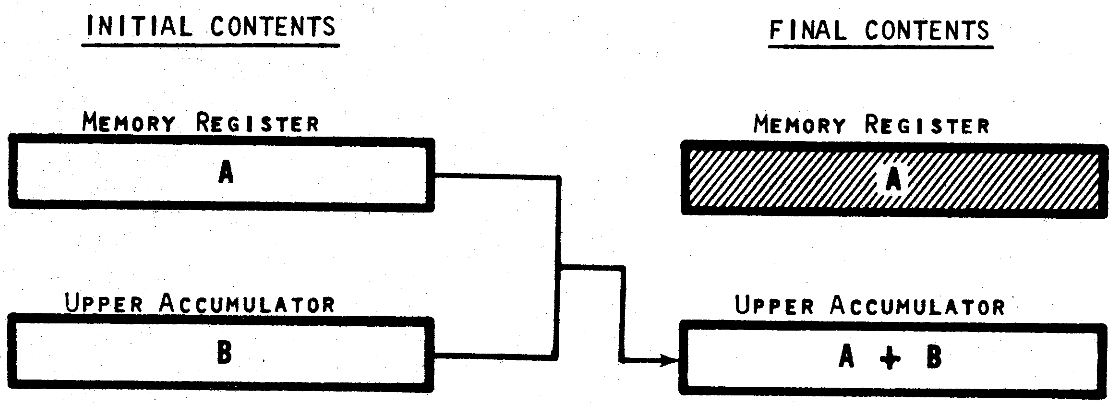
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Turns on Branch Control

Branch Control-------Turned on by overflow

Registers affected---Upper Accumulator



ADD TO LOWER ADL

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
29	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

ADDS THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY THE DATA ADDRESS TO THE CONTENTS OF THE LOWER ACCUMULATOR. RETAINS THE SUM IN THE LOWER ACCUMULATOR.

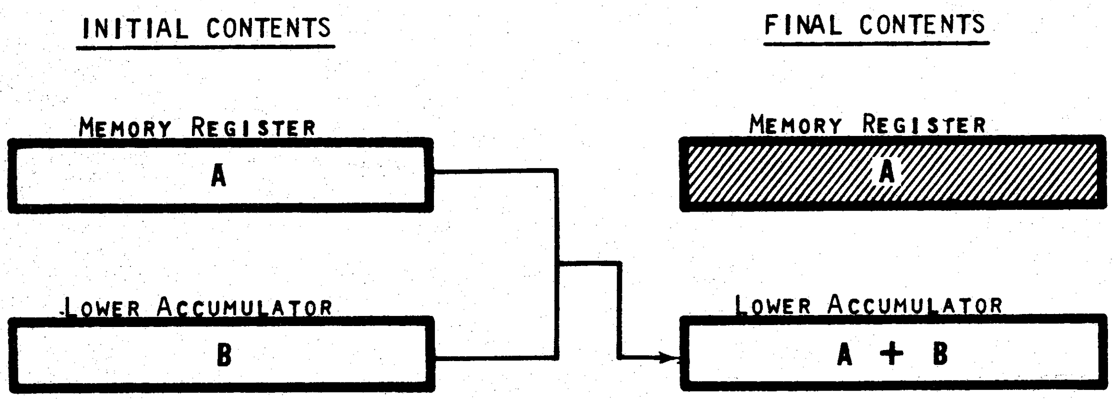
THE DATA ADDRESS MAY BE MODIFIED BY INDEXING.

MINIMUM TIME---------4 WORD TIMES

OVERFLOW-------------TURNS ON BRANCH CONTROL

BRANCH CONTROL-------TURNED ON BY OVERFLOW

REGISTERS AFFECTED---LOWER ACCUMULATOR



SUBTRACT FROM UPPER SBU

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
30	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

Subtracts the contents of the memory location specified by the Data Address from the contents of the Upper Accumulator. Retains the difference in the Upper Accumulator.

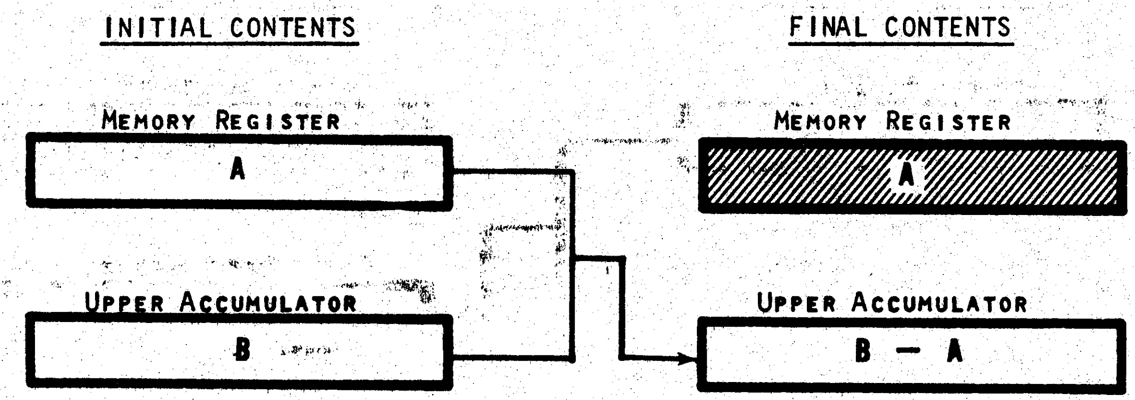
The Data Address may be modified by indexing.

Minimum time---------4 word times

Overflow-------------Turns on Branch Control

Branch Control-------Turned on by overflow

Registers affected---Upper Accumulator



SUBTRACT FROM LOWER **SBL**

INSTRUCTION WORD FORMAT

Command	D-Track	D-Sector	N-Track	N-Sector	X-Tag
31	(Data Address)		(Next Address)		0/1
0 4	5 11	12 17	18 24	25 30	31

SUBTRACTS THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY THE DATA ADDRESS FROM THE CONTENTS OF THE LOWER ACCUMULATOR. RETAINS THE DIFFERENCE IN THE LOWER ACCUMULATOR.

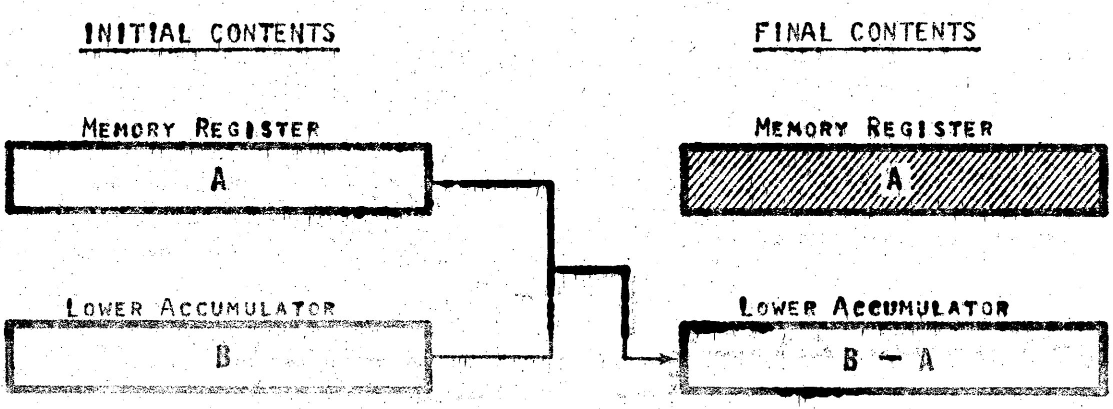
THE DATA ADDRESS MAY BE MODIFIED BY INDEXING.

MINIMUM TIME---------4 WORD TIMES

OVERFLOW-------------TURNS ON BRANCH CONTROL

BRANCH CONTROL-------TURNED ON BY OVERFLOW

REGISTERS AFFECTED--LOWER ACCUMULATOR



To RPC -4000 ADDENDUM Programming Manual

April 1960 Edition

Page 34 - If the D-track field of the INP instruction contains a value other than 000 or 064, the character which enters the Lower accumulator will be the logical product of the incoming tape character and the corresponding bits from the D-track field of the instruction word.

Page 39 - The description of the SLC instruction shoudl read as follows:

The value contained in the double length accumulator (combined Upper and Lower) is shifted to the left until bit 1 contains the first significant magnitude bit, or until the sum of the D-sector value of the instruction word and the number of bit positions shifted equals 64. Following the shift, the Lower accumulator is cleared to zero, and the sum of the D-sector value and the number of shifts is placed in Lower accumulator bits 12 through 17, modulo 64. That is, a sum of 64 will appear in the Lower as zero.

The Data address may be modified by indexing.

Page 41 - The MPT instruction may cause an overflow out of the left end of the affected accumulator but this overflow will not turn on Branch Control.

The addition of two diodes to the logic board can make this command function as a Shift Left1,2or3 bits in 4 word times . In this case The D-track value will be some other value than 000 of 064.