RCA ELECTRONIC DATA PROCESSING SYSTEM 501

PROGRAMMERS' REFERENCE MANUAL

RADIO CORPORATION OF AMERICA

Electronic Data Processing Division
Camden, New Jersey

RCA ELECTRONIC DATA PROCESSING SYSTEM 501

PROGRAMMERS' REFERENCE MANUAL

A General Introduction to the RCA 501 System
With Special Emphasis on the Computer

RADIO CORPORATION OF AMERICA

Electronic Data
Processing Division
Camden, New Jersey

The data herein presented is subject to minor change, without notice.

Supplements may be provided to advise of such revisions or additions.

First Printing, August 1958
Second Printing, November 1958

PREFACE

This manual is presented as a programmers' description of the RCA 501 System. As such, it emphasizes the Computer proper and includes only a general functional description of the associated peripheral devices in the RCA 501 System. Advanced programming techniques, operating procedures, and system analysis are not within the scope of this manual, but will be the subject of separate publications.

Particular emphasis has been placed on the individual Computer instructions, to which the largest portion of this manual is devoted. For ready reference in programming, a summary of the instructions, including register settings and automatic functions, and a summary of instruction timing appear in the appendices. A glossary of terms and a list of abbreviations used in the text are also appended.

CONTENTS

ILLUSTRATIONS

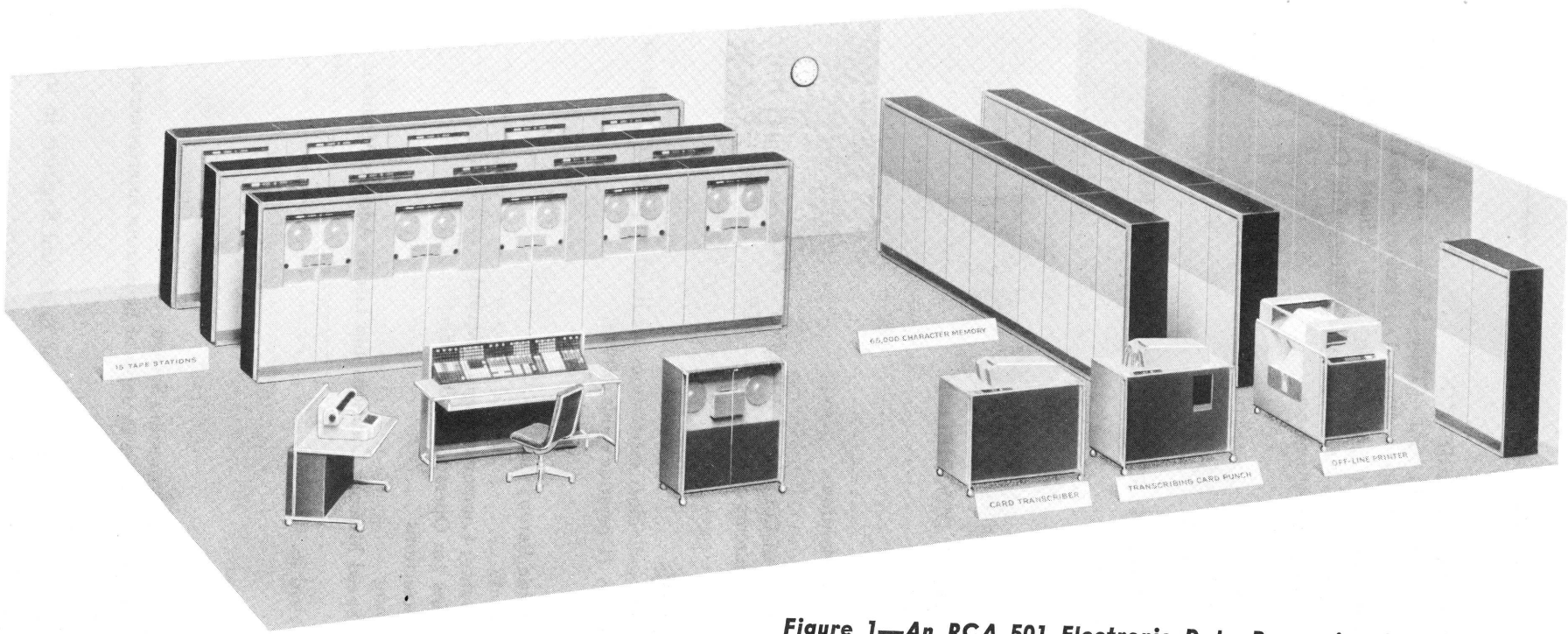


Figure 1—An RCA 501 Electronic Data Processing System

THE RCA 501 SYSTEM

GENERAL DESCRIPTION

The RCA 501 ELECTRONIC DATA PROCESSING SYSTEM is a general purpose, self-checking, readily expandable system, in the intermediate and large-scale performance class. The System incorporates extremely advanced logic, team-developed by engineers and programmers, and based upon extensive study of the characteristics of commercial data. All-transistor logic and printed circuitry contribute to attainment of the original design objectives of combining the highest performance level with the lowest possible cost of equipment and maintenance.

The RCA 501 is a complete system, capable of handling alpha-numeric data with magnetic tape, punched paper tape and punched card input and output, as well as printed output. In addition, drum storage devices provide fast random access, bulk storage for data and programs.

System efficiency is enhanced by:

- completely variable data organization, which conserves space on tape and in the internal memory and decreases processing time;
- four-character parallel transfer;
- increased data transfer rates;
- addressable registers;
- built-in and programmed accuracy controls, checking correct transfer of data in peripheral devices, into and out of the Computer, and within the Computer proper; additional controls ascertain correct arithmetic operations;
- time-shared electronics, permitting simultaneous operation of input-output devices with Computer functions;
- ready expandability of size of the internal memory and the type and number of peripheral devices.

The Computer is a general purpose, digital, sequentially controlled, random access, transistor machine, consisting of a number of integrated units (and attendant power supply): High-Speed Memory, Program Control, Tape Selecting and Buffer Unit, and Console with an associated Monitor Printer and Paper Tape Punch, and a Paper Tape Reader.

The **High-Speed Memory** is a random access, magnetic core device which provides storage and work area for programs and data. The memory is available in increments of 16,384 character locations and may be expanded to a maximum of 262,144 locations. Each location is individually addressable and can store any one of the sixty-four RCA characters. These characters (RCA 501 Code) include all the letters of the alphabet, the ten decimal digits, control symbols and special marks. One character or four characters in parallel can be addressed, brought into the Memory Register and regenerated in their original locations in one 15-microsecond cycle.

The **Program Control** is the arithmetic and logical control element of the Computer. It interprets and executes the instructions of the program stored in the High-Speed Memory and performs the automatic accuracy checks. The Computer and the on-line peripheral devices operate in accordance with a stored program of two-address instructions. The instructions that can be executed by the Program Control include all the categories necessary for processing of data: Input-Output, Data Handling, Arithmetic, and Decision and Control. Each instruction is made up of eight RCA characters and consists of four parts: (1) an operation code (read, multiply, transfer, etc.), (2) an A address (usually the High-Speed Memory address of an operand or the left boundary of an operand), (3) a B address (usually the High-Speed Memory address of an operand or right boundary), and an N code. The N code permits automatic modification of address A and/or B through the use of any of the seven (four static and three dynamic) Address Modifiers.

The **Tape Selecting and Buffer Unit-A** permits connection of one to eight Tape Stations to the Computer, and controls reading from and writing to magnetic tape on these stations. Program instructions designate the appropriate Tape Station for input or output. The number of Tape Stations directly controlled by the Computer may be increased to as many as sixty-three by connection of a **Tape Selecting Unit-B** to each of the Unit-A trunk lines. Write-out from the Computer to magnetic tape is at the rate of 16,667 or 33,333 characters per second. Read-in from magnetic tape is at the rate of up to 33,333 characters per second.

The **Console** provides for complete monitoring of operation of the Computer and the on-line devices, with panel display and control of register and status level action. Automatic and manual operation, maintenance, program insertion and program testing can be accomplished from the Console.

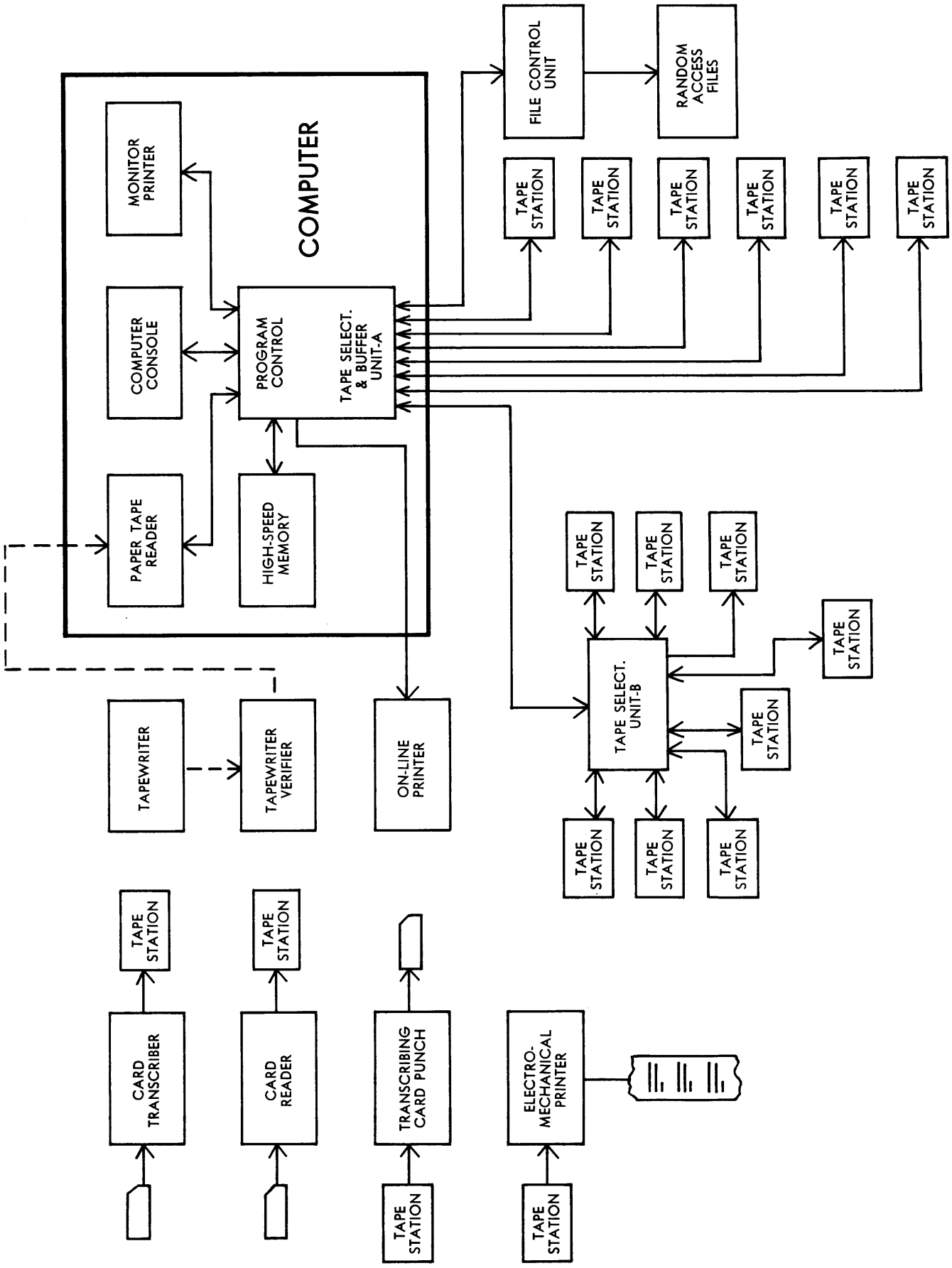


Figure 2—The RCA 501 System. Diagrammatic Example of Equipment Interconnection

Console facilities include:

1. Manual start and stop at a given instruction or at a given address.
2. Control and display of registers and counters.
3. Accuracy-checking indicators.
4. Indicators for currently performed instruction.
5. Indicators for last or currently selected Tape Station.
6. Control and display of Character Recognition flip-flops.
7. Breakpoint switches.
8. Alarm indicators.

The **Monitor Printer** is an on-line device, similar to an electric typewriter, that prints on paper stock from information received directly from the Computer's memory. It operates at the rate of ten characters per second and is used primarily for program operational control, program testing, and exceptional types of output. The Paper Tape Punch associated with this device can produce seven-hole punched tape simultaneously with the Monitor Printer's output of hard copy.

The **Paper Tape Reader** accepts seven-hole punched paper tape and operates at the rate of 400 characters per second. It is used largely for initial program insertion, program testing, "one-shot" programs, and insertion of periodically changing constants.

High-Speed printing can be accomplished on-line **(On-Line Printer)** or off-line **(Electro-Mechanical Printer)** in the RCA 501 System. The print line capacity is 120 characters and the print rate is 600 lines per minute. The On-Line Printer accepts data directly from the Computer memory and operates under the direction of the stored program. The Printer is converted to off-line by the addition of a **Data Editor,** operating then under the direction of a plugboard program and a paper tape loop, with information received directly from a Magnetic Tape Station.

Included in the peripheral equipment complement for the RCA 501 System are a Card Transcriber, Transcribing Card Punch, Tapewriter, Tapewriter-Verifier, and Random Access File. These devices are only briefly described here, since they will be fully detailed in separate manuals.

The **Card Transcriber** is comprised of two units, a **Card Reader** and a **Card Editor.** The Card Reader may be used without the Editor, in which case editing is reserved for the Computer. This device converts characters on eighty-column punched cards to RCA coded characters on magnetic tape, at the rate of up to 400 cards per minute. The Card Reader includes a control panel, an automatic card-handling mechanism and two card-reading stations. Each card is read at both stations, and the readings are compared as an accuracy check. The Card Editor permits rearrangement and selective transcription of card data and insertion of additional characters. The Card Transcriber employs transistor circuitry and accuracy checking, including parity, comparison and multi-punch checks.

The **Transcribing Card Punch** converts RCA coded characters on magnetic tape to punched card code on 80-column cards at the rate of 150 cards per minute. This device employs transistor circuitry and includes a card-punching unit and an electronics unit with a control and indicator panel. The card-punching unit is comprised of an automatic card-handling mechanism, a card-punching station and a card-reading station. A manually wired plugboard is used to rearrange information, to provide for overpunching, and to insert special control symbols and additional characters. Accuracy controls include parity, correct data format, and correct punching checks.

The **Tapewriter** and **Tapewriter-Verifier** are used for original preparation and verification of RCA coded, seven-hole punched paper tape for subsequent input to the Computer via the Paper Tape Reader. These devices are keyboard operated and simultaneously print on paper stock the same information that is being punched on tape. The Tapewriter-Verifier automatically checks the accuracy of its output by comparison with a previously prepared (Tapewriter) punched paper tape. Whenever a character being punched on the Tapewriter-Verifier is not in agreement with the related character on the original tape, both the keyboard and the punch lock. Both devices will function at typing speeds up to 10 characters per second, and both include parity checking.

The **Random Access File** is a drum storage device which provides fast random access, bulk storage for data and programs and operates under automatic program control. Each unit has a capacity of at least 1.5 million characters, with 192-millisecond average random access time to any data (167 milliseconds for drum access plus 25 milliseconds for relay switching between tracks).

There are 300 tracks on the drum. The capacity of each track is 5000 alpha-numeric characters. The character bits are recorded serially in each track around the drum. Bits are recorded or read at 6.7-microsecond intervals (character transfer rate, up to 18,700 per second).

The drum code is eight-bit serial, with four "1" and four "0" bits per character. Conversion between this and the RCA 501 Code is handled automatically within the File Control Unit.

The **File Control Unit** used in conjunction with the Random Access File enables the Computer to control

and to read and write information on a maximum of twelve Random Access Files, in accordance with the Computer program. The File Control Unit can be connected to the Tape Selecting and Buffer Unit-A or B in the same manner as the Tape Station. Reading or writing of information on a Random Access File may be accomplished simultaneously with other Computer functions. Simultaneous operation between the Computer and two Random Access Files is possible if each File is connected to a separate File Control Unit.

Summary of Peripheral Equipment Performance

On-Line

Input:

Paper tape	1000 characters per second
Magnetic tape	up to 33,333 characters per second
Random Access File	18,700 characters per second

Output:

Magnetic tape	16,667 or 33,333 characters per second
Paper Tape Punch	10 characters per second
Monitor Printer	10 characters per second
On-Line Printer	600 lines per minute, 120 characters per line
Random Access File	18,700 characters per second

Off-Line

Input:

Card Transcriber	400 cards per minute
Tapewriter	10 characters per second
Tapewriter-Verifier	10 characters per second

Output:

Electro-Mechanical Printer	600 lines per minute, 120 characters per line
Transcribing Card Punch	150 cards per minute

ACCURACY CONTROL

Automatic accuracy controls incorporated in the RCA 501 System are designed to prevent incorrect information from entering or leaving the System and are selectively engineered to avoid overloading the System with checks at non-critical points.

Accuracy Checking Techniques

Parity Checking. Each character on tape carries an extra bit to make up an odd number of "1" bits on magnetic tape and an even number on paper tape. Correct parity is ascertained on read-in, during data flow in the Computer, and on write-out. Parity checking in the 501 System is well illustrated in reading and recording of data on magnetic tape. Each character transmitted by a user device (Computer, Card Transcriber, etc.), is checked for parity. On receipt of the character at the Tape Station, where it is recorded in duplicate, echo returns from the recording heads are parity checked in the Tape Station, and the user device is notified that the character was recorded successfully. The user devices also check parity on data received from tape.

Dual Recording on Magnetic Tape. The bits of each character plus a timing bit are recorded in duplicate, in sixteen channels, across the width of the tape. All dually recorded bits of the character are read or written simultaneously. Either one of the two recorded spots for a single bit may be missing, and the bit can still be read. In addition to its value as an accuracy control measure, dual recording lengthens tape life.

Automatic Rerun (Rollback). If a parity error does occur on a magnetic tape "read," the tape is automatically backed up and the read instruction is re-executed. The Computer stops, with Console indication of the reason for stoppage, if an error is detected on re-reading. (See Appendix I for description of Rollback.)

Arithmetic Accuracy Checking. Each arithmetic cycle of an arithmetic operation is performed twice, first with the original operands and then with the complements—all during the same time cycle—and the results are compared for agreement.

Application of Accuracy Checking Techniques

Program Control. The following Program Control conditions cause the Computer to stop:

1. Incorrect parity in Memory Address Register.
2. Incorrect parity in Memory Register.
3. Arithmetic Unit malfunction.
4. Incorrect parity on output of bus adder.
5. Illegal operand in decimal operation.
6. Incorrect parity in Normal Operation Register.
7. Incorrect transfer of operation from Normal to Simultaneous Mode.
8. Malfunction of previous result indicator.
9. Time pulse generator malfunction.

Input-Output. The following input-output conditions cause the Computer to stop:

1. Tape Station reading extra bits in Intermessage Gap.
2. Missing clock pulse when reading from Tape Station.

3. Missing clock pulse when writing out from Computer.
4. Tape Station does not obey control signals.
5. Odd number of characters from paper tape block read.
6. Second parity error in tape read (see Automatic Rerun).
7. Incorrectly selected tape.
8. Incorrect data format (incorrect Start Message-End Message sequence).
9. Incorrect parity at the output of Computer write buffer or of the Tape Station writing head.
10. Incorrect paper tape parity.
11. On-Line Printer not operable.
12. On-Line Printer paper supply low.

THE RCA 501 NUMBERING SYSTEM

The RCA 501 System employs a binary numbering system to represent information, utilizing seven binary digits, or bits, to represent each RCA character (see Organization of Data on Tape, page 6, and the RCA 501 Code, Appendix II).

The following table lists binary equivalents of decimal numbers as they might appear in a theoretical computer that employed, not a constant number of bits to represent each character, or decimal digit, but as many as a given decimal number required.

A. Partial Table of Decimal and Binary Equivalents

Decimal	*Binary*
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
32	100000

Binary-coded decimal representation employs four binary digits, or bits, to represent each decimal digit, e.g., decimal zero is represented as 0000, decimal 3 as 0011, etc. *Excess-3 binary-coded decimal representation* adds 3 to each decimal digit and employs four bits to represent the resulting digit. An excess-3 binary-coded representation of decimal zero, then, looks like a binary-coded decimal 3; an excess-3 representation of decimal 6 looks like a binary-coded decimal 9 (Table B).

B. Table of Decimal, Binary-Coded Decimal, and Excess-3 Binary-Coded Decimal Equivalents

Decimal	*Binary-Coded Decimal*	*Excess-3 Binary Coded Decimal*
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Of the seven bits that make up each of the sixty-four RCA characters, the leftmost is the *parity* bit, used for accuracy checking; the remaining six are the *information* bits. The positional significance of the information bits increases from right to left (from 2^0 to 2^5).

bit position	P	2^5	2^4	2^3	2^2	2^1	2^0
bits (x = 0 or 1)	x	x	x	x	x	x	x

The rightmost four information bits ($2^0 - 2^3$) constitute the *arithmetic* portion of a character. (Table B shows only the arithmetic bits.) The RCA Code is so devised that the leftmost two information bits (2^5 and 2^4) are the same (01) for all decimal digits (0 through 9); this permits the Computer to consider only the arithmetic portion of characters in decimal arithmetic operations (add, subtract, multiply, divide). Further, the arithmetic portion of each decimal digit is excess-3 binary-coded decimal. The particular advantages here are (1) ease of complementation (note, in Table B, the reverse bit configuration for 0 and 9, 1 and 8, 2 and 7, 3 and 6, 4 and 5), and (2) a carry is propagated in addition of two excess-3 binary coded operands whenever a carry would be propagated by addition of their decimal equivalents.

RCA 501 programs are written in *octal notation;* this applies to data, addressing, and all instruction components. Octal notation serves as a mnemonic device for reading and writing binary representations of RCA characters (see Octal Equivalents in the RCA 501 Code, Appendix II). To read a binary code as octal:

1. Ignore the parity bit and divide (visually) the information bits into two 3-bit groups;
2. Read the two groups as one octal number by deriving the octal digit for each group.

Each group of three bits has a corresponding octal digit as follows:

Bits	*Octal Digit*	*Bits*	*Octal Digit*
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

For example:

1. The bit configuration for the letter "T" is 1110011. Ignoring the parity bit and dividing the information bits, one can easily see this as 110 011 and read the octal equivalent as 63.
2. The bit configuration for decimal digit "9" is 0011100.

bit position	P	2^5	2^4	2^3	2^2	2^1	2^0
bits	0	0	1	1	1	0	0
octal equivalent			3			4	

[34 = octal equivalent of decimal 9, or $(9)_{10} = (34)_8$].

Since internal addressing is octal and programs are entirely written in octal notation, only the digits 0 through 7 will ever appear on an RCA 501 Computer Program Record (see Appendix III).

ORGANIZATION OF DATA ON TAPE

General

The seven-bit binary code for each RCA character is listed in Appendix II. In the RCA 501 Code, the parity bit (P) is chosen to be a "one" or a "zero" so that every character will contain an even number of "one" bits on paper tape and an odd number on magnetic tape. An eighth bit is recorded with each character on magnetic tape for timing.

Definitions

Bit. A bit is a single binary digit, having a value of either 0 or 1.

Character. An RCA 501 character consists of six information bits and one parity bit combined to represent a decimal digit, a letter of the alphabet, a punctuation or other special mark, or a control symbol (see The RCA 501 Code, Appendix II).

Item. An item consists of such characters as are necessary to specify a particular unit of information (a numerical quantity, an alphabetic name, a street address, a stock number, etc.). An item is preceded by an Item Separator symbol (ISS).

Message. A message consists of a Start Message symbol (SM); one or more related items, each preceded by an Item Separator symbol; and an End Message symbol (EM), in that order.

Block. On magnetic tape, a block consists of eight or more characters, preceded and followed by an Intermessage Gap. Intra-block blanks must constitute less than 75 microseconds of tape time.

On paper tape, a block consists of an even number of characters equal to or greater than sixteen, without intra-block blanks; it is preceded and followed by an Intermessage Gap. (Corrective overpunching, to delete a character, is ignored in this character count.) The characters must represent only the decimal digits 0 through 7 (octal 23 through 32). (See discussion of decoding circuitry, Paper Tape Reader, page 10.)

Blocks on magnetic or paper tape are read and written without regard to message structure rules—they are delineated by Gaps, rather than by control symbols, and need not contain any control symbols.

Line. A line is composed of the characters from a single message which may be read from magnetic tape into the Electro-Mechanical Printer during a single read cycle. Each such line is terminated with an End Message (EM) or Line Shift (LS) symbol. The LS symbol appears only as the last character of a line. When a Page Change (PC) symbol or an Item Separator symbol is used to provide an additional paper control symbol, it must always be followed by LS or EM and then the Intermessage Gap. Also, either EM or LS preceded and followed by an Intermessage Gap is a line and must be treated as such on tape. (In the Electro-Mechanical Printer, the maximum number of characters that can actually be printed is 120 per line. Each character space to appear in the printed line decreases the 120 maximum by one. On tape, a line may include non-print characters and control symbols in addition to the 120.)

File. A file consists of any number of related information units, in message or block format; it may consist of several tapes (reels) or any part of one tape. A file is terminated by an End File (EF) symbol, preceded and followed by an Intermessage Gap.

In a multi-tape file, all but the last tape are terminated by an End Data (ED) symbol alone, preceded and followed by an Intermessage Gap. The end of file on the last tape is indicated by an EF, preceded and followed by an Intermessage Gap. If this is a full tape, or a partially filled tape with no other valid information following the file data, an ED follows the EF, separated from it and followed by an Intermessage Gap.

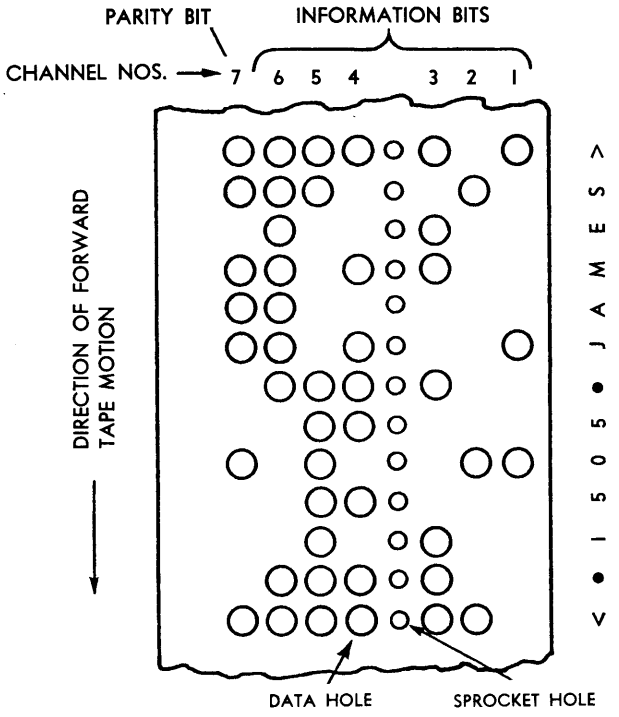


Figure 3—Recording on Paper Tape

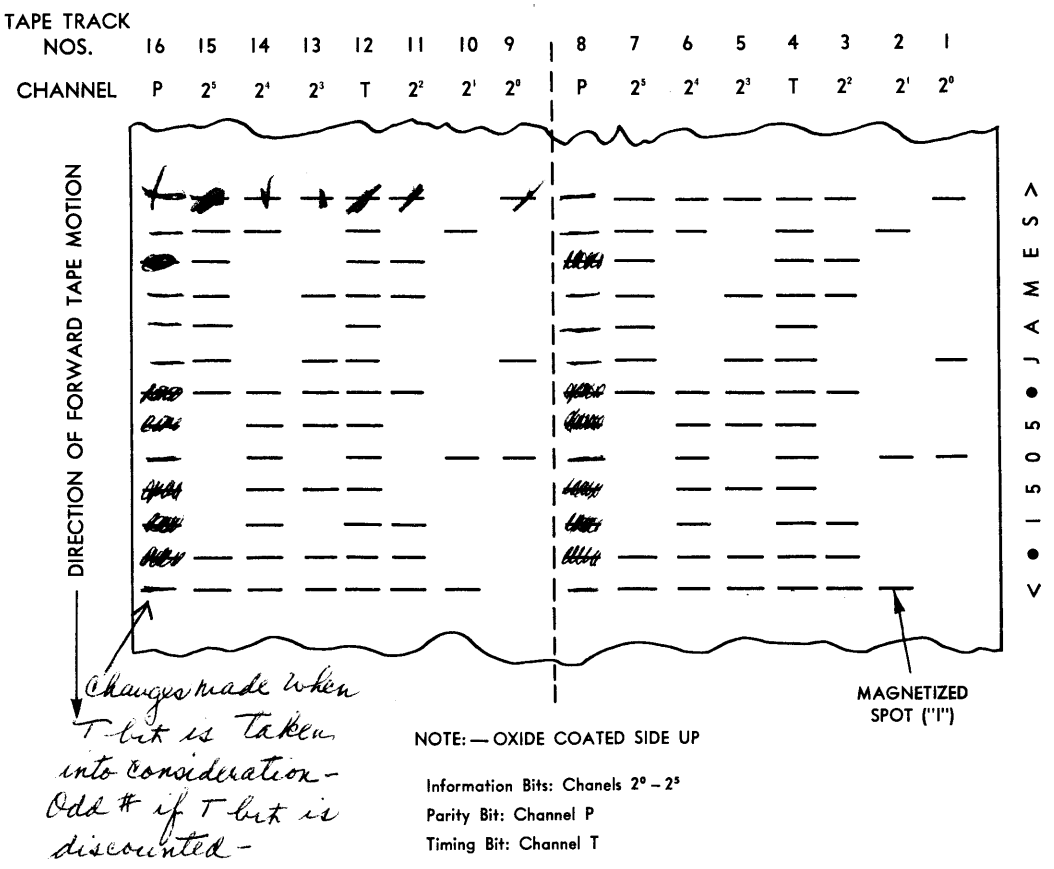


Figure 4—Recording on Magnetic Tape

If more than one file appears on a tape, each file except the last is terminated by an EF only. The last complete file on a tape is terminated by an EF followed by an ED.

ED and EF are each preceded and followed by an Intermessage Gap. They are never accompanied by Start and End Message symbols, and they may never appear within a message.

If a file is read or written in block format, the ED and the EF are treated as separate blocks. If a file is read or written in message format, the ED and the EF are treated as separate messages.

Intermessage Gap. An Intermessage Gap is a length of unrecorded tape sufficient to allow for Gap detection and stopping and starting of the tape. This is a minimum of 0.34″ on magnetic tape (for block or message format). On paper tape, it is a minimum of three character positions for message format and a minimum of four character positions for block format.

Arrangement of Data on Tape

Arrangement of Bits to Form Characters. Figure 3 illustrates the arrangement of information on paper tape. There are seven data hole positions per row. Each row represents a character. The presence of a punched hole represents a "one" bit, and the absence of a hole represents a "zero" bit. The positions of the bits are numbered 2^0 through 2^6, corresponding to the information and parity bit positions of an RCA binary coded character. A row with all seven channels (plus an extra, eighth, channel) punched indicates that the character in this row has been deleted; such punching does not represent an RCA 501 character (see The RCA 501 Code, Appendix II).

Bits are recorded on magnetic tape as magnetic spots in rows across the tape (Figure 4). Each bit is written in two locations as an accuracy control measure, giving 16 tracks across the width of the tape (including the timing bit). Each of the two locations is capable of producing a standard signal. Thus, either one of the two recorded spots for a single bit of information may be missing, and the bit can still be read.

Arrangement of Characters to Form Items. All characters are recorded on tape serially so that the characters making up an item follow one another in sequence from most to least significant. Each item is preceded by an Item Separator symbol.

Each item of a message may have variable length. Inclusion of the Item Separator symbols allows the use of variable length items and the omission of items, without changing the positional significance of any item in the message (see Variable Item and Message Length, this page).

Arrangement of Items to Form Messages. The items of a message follow one another in sequence, each being preceded by an Item Separator symbol. In every message of a given type, the *n*th item always has a given connotation. Therefore, a count of the Item Separator symbols, starting from the first or from a program-oriented point in a message, permits location and identification of any item.

In the event that a particular item is omitted, the Item Separator symbol can be recorded on tape in its proper sequence when it is necessary to preserve the positional significance of the items that follow. However, the Item Separator symbol for an omitted item may also be omitted if there is no valid information following it in the message. In this case, the End Message symbol follows immediately after the last item present. This avoids writing an unnecessary number of consecutive Item Separator symbols at the end of a message.

All messages consist of a Start Message symbol followed by an Item Separator symbol and the characters of the first item, the succeeding items (each preceded by an Item Separator symbol), and an End Message symbol, in that order. The SM and EM symbols appear only in the positions defined herein.

Miscellaneous

Equipments that record data on magnetic tape provide for erasing a minimum length of 1.35″ at the beginning of each tape before recording. The equipment recognizes the beginning of the tape by a Beginning of Tape Level which is generated by a permanent indicator (Beginning of Tape Control) in a fixed position on the magnetic tape (not an RCA 501 character).

An end of tape warning (ETW) device is provided to indicate that approximately 5 feet of usable magnetic tape are available; this permits recording of an entire message and an End File and/or End Data symbol after the warning is received. This warning is not an RCA 501 character, but is a signal generated by a permanent indicator in a fixed position on the tape.

Variable Item and Message Length

Data storage in the RCA 501 System incorporates *true variable* item length. This concept may be more fully appreciated if prefaced with a discussion of *fixed* and *fixed variable* word and block length.

"Word" is generally defined as a fixed number of consecutive characters or character locations, and "block" as a fixed number of consecutive words, in primary or secondary storage. These terms, word and block, are more aptly used with respect to *fixed* and *fixed variable* systems, but are not particularly applicable to a *true variable* system.

In a computer system employing fixed word length, the number of characters per word and the number of words per block are characteristic of the system, incorporated in the circuitry. A computer with a fixed word length of 12 characters dictates the use of some multiple (not fraction) of 12 for each and every item (employee number, name, pay rate, etc.), in a message and a fixed number of words for each message in a file. If the employee number is made up of five digits, the word in which this item was stored would be filled out with redundant zeros or spaces (e.g., +64398000000). This would be true for each employee number in the file. The alternative of utilizing these zero-filled positions by packing more than one item into a word entails additional program instructions, with consequent increased processing time, and is possible only to a limited degree. Even with packing of words, the fixed block size may entail zero-filling of one or more entire words at the end of each block.

In a fixed variable (non-standard maximum item length) system, the number of character positions for each item is assigned in accordance with the anticipated maximum for that item. In such a system, then, item (data field) layout is analogous to that used for punched cards. These lengths may be individually predetermined for each file, but remain constant for each message in the file. For example, in an inventory file, cost per unit might vary from two cents for one stock number to some five-digit figure for another. This maximum would dictate that five character positions be used even where there is only one significant digit; two cents might then be written as 00002. Stock numbers in this file, however, could be assigned a different number of character positions, stock description still a different number, and so on. Thus, fixed variable word length provides greater flexibility than does fixed word length.

Data storage in a *true variable* item length system does not have the limitations imposed by fixed or fixed variable systems. In the RCA 501, the use of control symbols and the ability to address each character location individually permits the length of any item in any message to be in strict accordance with that item's actual character count. This allows for total variability of item and message length, but does not preclude the use of fixed or fixed variable lengths when the programmer finds this expedient.

In each of these categories—fixed, fixed variable, and true variable—the method of internal storage is extended to external storage; when redundant zeros or space characters are required to fill out a word in the computer memory, they are also carried on tape. With a given tape density (number of characters packed per inch) and a given tape speed (number of inches per second), a characteristic business file would require less tape footage and could be read and written out in less time when true variable item length is utilized.

ON-LINE PERIPHERAL EQUIPMENT

FUNCTIONAL DESCRIPTION

PAPER TAPE READER

The Paper Tape Reader associated with the RCA 501 System is used to transcribe incoming data to magnetic tape, to enter data initially into the Computer during program testing, to transcribe programs, and to enter into the Computer periodically variable constants for production programs (e.g., today's date).

The Paper Tape Reader is photoelectric and operates at the rate of 1000 characters per second. It uses one-inch wide, seven-hole punched tape; the seven channels across the width of the tape correspond to the seven bit positions (six information bits and one parity bit) of an RCA character. Characters on punched paper tape have even parity. (See Figure 3.) When a character position on paper tape contains a punch in all seven channels, plus an eighth punch, this is interpreted as "delete character"; it is not an RCA character and no attempt is made to read it into the Computer. Information on paper tape may be in either message or block format.

Since the tape does not stop immediately after the last character is read in, but may glide the equivalent of three character positions, a Gap (unpunched tape) of three character positions is left between successive messages, and four character positions between successive blocks.

To facilitate the insertion of programs, a special method is used to block read paper tape. Programs are written, on the code sheet, in octal notation. An RCA 501 Tapewriter is used to create a punched paper tape from the code sheet, so that the program can be read into the Computer. One *decimal* key is depressed for every octal *digit* on the code sheet, thereby producing on the tape one RCA character (two octal digits) for every octal digit on the code sheet.

In a block read from paper tape, characters automatically enter decoding circuitry before they are transferred into the HSM. Decoding may be explained as follows: $(23)_8$ is subtracted (binary subtraction) from the first character on tape, and the rightmost three bits of the difference are stored as the *leftmost* three bits of the decoded character; $(23)_8$ is subtracted from the second character on tape, and the rightmost three bits of the difference are stored as the *rightmost* three bits of that decoded character. The combined result, with a parity bit generated, is then stored in the High-Speed Memory in the same fashion as are characters read from magnetic tape. The process is then repeated for each character on the paper tape until a Gap is recognized. On a block read from paper tape, then, the leftmost three bits (left octal digit) of each character stored in the HSM will be derived from the first, third, fifth, seventh, etc., character read in from the tape; and the rightmost three bits (right octal digit) will be derived from the second fourth, sixth, eighth, etc., character read in from the tape. When the last character on tape has entered the decoding circuitry, an alarm occurs if the count is less than sixteen, or an odd number.

This process may be represented as follows:

Instruction characters as they appear on code sheet:

0	A	N	B
05	014103	00	600000

Octal digits punched on paper tape	23	30	23	24	27	24	23	26	23	23	31	23	23	23	23	23
Automatically subtracted in decoding circuitry	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
Difference	00	05	00	01	04	01	00	03	00	00	06	00	00	00	00	00
Decoded characters stored in HSM	05		01		41		03		00		60		00		00	

This decoding occurs only when paper tape is read in block format, permitting the digits 0 through 7 (octal 23 through 32) to be properly decoded. Block format on paper tape is largely restricted to programs for initial insertion into the Computer. Data (as against program instructions) on paper tape are punched and read in message format, in standard RCA 501 Code, and are not subjected to code conversion.

The following formula is used for paper tape read time:

Total no. of char. $\div$ 400 = total time in seconds

Start time is negligible for paper tape and is not considered a time factor.

MONITOR PRINTER

The Monitor Printer associated with the Computer Console prints out information from the High-Speed Memory of the Computer under the direction of the stored program at the rate of ten characters per second. Two modes of operation are possible:

1. *In Computer or program testing,* a character will be printed for every RCA character that might be stored in the designated area of the High-Speed Memory.

Carriage return and a line shift automatically occur after printout of an End Message symbol or when the right-hand paper margin is reached.

The table below shows the character that will be printed for each octal number in the High-Speed Memory [e.g., a space symbol, $(01)_8$, in the HSM will be printed out as an underline, $(23)_8$ will be printed out as a decimal zero, $(24)_8$ as a decimal one, etc.].

1st digit of octal no. \ 2nd digit of octal no.	0	1	2	3	4	5	6	7
0	a	Underline	‡	(	)	"	:	$
1	%	;	&	'	minus sign	*	.	h
2	q	r	/	0	1	2	3	4
3	5	6	7	8	9	,	#	x
4	A	B	C	D	E	F	G	H
5	I	J	K	L	M	N	O	P
6	Q	R	S	T	U	V	W	X
7	Y	Z	s	t	●	>	<	v

2. *For edited output,* printing of space symbols, Item Separator symbols and End Message symbols can be suppressed. The End Message symbol, though suppressed, will still effect a carriage return and line shift. A suppressed space symbol will cause a horizontal shift of one character position. The Monitor Printer can thus be used for output of documents (summary totals, cost distributions, etc.), utilizing either pre-printed or blank paper stock.

The **Paper Tape Punch** associated with the Monitor Printer is used for manual preparation of short paper tapes and for output of data from the Computer. Punching is activated by flipping the Punch Switch on the Monitor Printer to the ON position, in which case both punched tape and printed copy on the Monitor Printer are obtained. This mode of operation is subject to the restriction that printing of a lower-case letter (see Monitor Printer chart above) will cause the octal value of the corresponding upper-case letter to be punched in the tape. For example, if the character $(00)_8$ is read out of the HSM, the lower-case letter "a" will be printed (Monitor Printer) and $(40)_8$, the octal value of an upper-case "A," will be punched on tape. [Of course, if $(40)_8$ is read out of the HSM, an upper-case "A" will be printed and $(40)_8$ will be punched on tape.]

Monitor Printer output time (with or without paper tape output) may be computed on the basis of the number of characters involved (total no. of char. ÷ 10 = total time in seconds). Start time is negligible and is not considered a time factor.

ON-LINE PRINTER

The On-Line Printer in the RCA 501 System is an all-transistor device which prepares output documents, printing data directly from the High-Speed Memory of the Computer. Data editing is accomplished in the Computer, under the direction of the stored program. Line skipping is controlled by the Computer program, either directly or through a Tape Loop on the Printer Unit.

Two Computer instructions are directly associated with the On-Line Printer—one initiates printing and the other positions the paper for the next line of printing. The latter instruction may specify the number of lines the paper is to be advanced, or it may refer to one of two information channels in the Tape Loop on the Printer Unit. One channel (Vertical Tabulation) is referenced in order to advance the paper to specific lines within the confines of a page, and the other channel (Page Change) is referenced to move the paper to the start of a new page.

Maximum print capacities are 120 characters per line, 10 characters per horizontal inch, and 6 lines per vertical inch. One line is printed in 66.7 milliseconds. For single-line paper advance, the paper motion time is 30 milliseconds. Printing and paper motion time, then, total less than 100 milliseconds, permitting single-spaced printing at the rate of at least ten lines per second (600 lines per minute). When more than three lines are skipped at one time, however, the paper advance rate is at least 50 lines per second.

Paper stock may be single or multiple sheet fanfold, from 3 to 22 inches in width and up to 17 inches in sheet length. One original plus up to three carbons (11-pound paper and 7½-pound carbon) may be used. Hecto and Multilith master stock may also be used.

Fifty-one RCA characters can be printed by the On-Line Printer. These include the 10 decimal digits, the 26 letters of the alphabet, and the following punctuation marks and symbols:

,	comma	*	asterisk
;	semicolon	&	ampersand
:	colon	/	virgule
.	period	%	per cent
'	apostrophe	$	dollar sign
(	open parenthesis	#	number sign
)	close parenthesis	—	minus sign
"	ditto or quotes		

The occurrence in an HSM print-out sector of an RCA character other than one of the fifty-one listed above will leave a blank in the related position in the printed line, with the exception that $(00)_8$ will result in an overprinting of the three symbols =, + and @. (These

symbols, though present on the print wheel, normally have no RCA Code equivalents.)

For accuracy control, a Printer Unit Inoperable alarm, which stops both the Printer and the Computer, is incorporated. Also, the Printer, by means of a microswitch, can sense a "low paper" condition and send a warning signal to the Computer. When the Computer program calls for Tape Loop activation of a page change and a "low paper" signal is present, both the Computer and the Printer stop, after accomplishment of the page change, to permit replenishment of the paper supply. Thus, printing on a page is completed before the operation is stopped.

MAGNETIC TAPE STATION

The Magnetic Tape Station is a fully automatic device, with transistor circuitry, which performs reading, writing and erasing operations on ¾-inch wide Mylar base magnetic tape, under control of the user equipment. On each Tape Station, two 10½-inch reels are mounted —a full reel and a take-up reel. The capacity of a reel is 2400 feet of tape, providing a minimum of 2300 feet of usable tape. Tape Station design facilitates manual interchange of reels, which can be accomplished in less than one minute.

The Tape Station can be instructed to move the tape in a forward or reverse direction. It can be directed to move the tape with or without writing. It can read the tape with or without transferring characters into the High-Speed Memory. It can be instructed to read all of the data serially or search for specified symbols. The Tape Station can, in response to one instruction, unwind the tape to the end, or rewind it to the beginning. Writing on magnetic tape is in the form of significant configurations of magnetic spots (see Organization of Data on Tape and Figure 4).

The Tape Station writes at the rate of up to 33,333 characters per second (33.3 KC). It accomplishes this by writing 333.3 characters to the inch while moving the tape at a speed of 100 inches per second. Information on magnetic tape is read at this same rate. It takes only 30 microseconds to read a character on tape into the High-Speed Memory, or to write a character from the memory onto magnetic tape.

Gaps on magnetic tape between messages or blocks are 0.34 inch. Part of the gap is attributable to the fact that tape glides for a short distance after being commanded to stop. On a tape write, 3.5 milliseconds elapse between the tape start command and write-out of the first character from the High-Speed Memory. The same time lapse occurs on a tape read. A lapse of 4.5 ($\pm$0.9) milliseconds occurs when a read-reverse tape instruction immediately follows a write instruction for the same tape. There is otherwise no appreciable delay in switching the direction of tape movement or in switching between reading and writing.

Up to 63 Tape Stations can be directly addressed by the Computer. Eight Tape Stations can be connected to Tape Switching and Buffer Unit-A. As additional Tape Stations are required in an installation, a Tape Switching Unit-B can be substituted for each of the original eight Tape Stations. The addition of one such unit will, therefore, permit as many as 15 Tape Stations to be connected (seven original and eight added) to the Computer. One A Unit and eight B Units are required to connect sixty-three Tape Stations to the Computer. Each Tape Station has a unique (octal) address, $(00)_8$ — $(76)_8$. The 64th address, $(77)_8$, is reserved for the Monitor Printer and the Paper Tape Punch and for the Paper Tape Reader.

The first eight Tape Stations (attached to Unit-A) are Computer identified by the left digit of the tape selection number (00, 10, 20, 30 . . . 70). With only these eight Tape Stations in a system, then, any number from 10 to 17 will select Tape Station 10; 30 to 37 will select Tape Station 30, etc. Each Tape Station connected to a B-type unit must be addressed by its individual Tape Station number. For instance, if Tape Stations are connected to 30, 31, 34, 35 and 37 of the 30–37 octet, and 32, 33 or 36 is addressed, a "non-operable" alarm will occur and the Computer will stop.

Tape Station accuracy controls include manual lockout and "write interlock." Manual lockout is provided on each Tape Station to insure safe manual operation procedures. When the Lockout Switch of a Tape Station is in the ON position nothing can be written on the tape at that Station. A "write interlock" feature safeguards reference tapes from human error by preventing writing (and erasing).

FUNCTIONAL DESCRIPTION

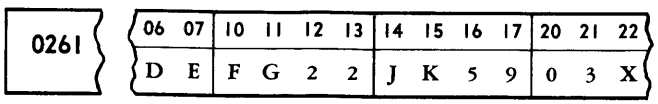
HIGH-SPEED MEMORY

The High-Speed Memory (HSM) in the RCA 501 System is constructed in modules. A module is made up of twenty-eight 64 x 64 matrices of magnetic cores. Each core can represent one bit; a matrix of cores, then, represents 4,096 bits and a module represents 114,688 bits. Since an RCA character is comprised of seven bits, each module can store 16,384 characters in 16,384 individual character locations.

The HSM is expandable from one to sixteen modules (four banks, each with a capacity of 65,536 characters), or up to a maximum of 262,144 character locations. Each location in memory has a unique address, consisting of three RCA characters (= three octal numbers = six octal digits), so that each location, or the contents thereof, is individually addressable. Though somewhat oversimplified, the HSM may be pictured as a rectangular array of locations, with the smallest address in the upper left-hand corner and the largest address in the lower right-hand corner. The lowest address in the HSM is always $(000000)_8$. Since the addressing scheme in the RCA 501 employs the octal number system, the highest address in a one-module memory is $(037777)_8$ and the highest address in a sixteen-module memory is $(777777)_8$.

To decrease processing time, the HSM is constructed so that four characters (twenty-eight bits), in four consecutive character locations, can be accessed in a single memory cycle. The Computer memory cycle is 15 microseconds; this means that characters may be addressed, brought into the Memory Register, and regenerated in their original locations every 15 microseconds.

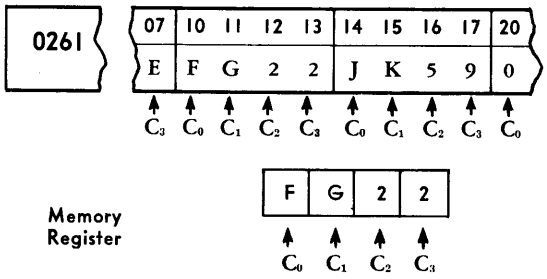
Each of these groups of four locations, or the contents thereof, is called a *tetrad*. A tetrad begins in a location addressed $(-----0)_8$ or $(-----4)_8$ and ends in a location addressed $(-----3)_8$ or $(-----7)_8$, respectively. In diagrammatic representations of portions of the HSM throughout this manual, tetrads are delineated by heavy vertical lines, as shown below.



The HSM address of each location shown in the above diagram consists of the two octal digits in the upper portion preceded by the four digits at the left (026106, 026107, etc.). The characters stored in these locations are shown in the lower portion of the diagram.

As stated previously, each character location has a unique address. A *tetrad address*, however, may be the address of any one of the four locations comprising that tetrad. No matter which one of the four is addressed, the contents of the entire tetrad will be brought into the Memory Register. Depending on the instruction being executed, since some RCA 501 instructions deal with characters singly, and some by tetrads—all four characters in the tetrad, the rightmost three, or only the character from the specified location will be processed.

For convenience in referring to the individual locations or characters within a tetrad, when the specific address is not pertinent, the symbols C_0, C_1, C_2, and C_3 are used. These symbols apply to any tetrad in the HSM and to characters in the Memory Register.



The primary purpose of the HSM is the storage of programs and data. These may be stored in any area of the memory, except that the first 164 locations (000000–000243) are reserved as *Standard HSM Locations* (see Appendix VII). This reserved area stores data used for address modification, special accuracy routines, and data for which special counters would otherwise be required.

THE BASIC INSTRUCTION

Instruction Format

Each of the forty-seven RCA 501 instructions consists of four parts, in the following order:

1. Operation Code (a one-character code for add, subtract, item transfer, etc.)
2. A Address (three-character HSM address of the augend, minuend, original location of an item, left boundary of a sector, etc.)

3. N Character (one-character code that can call for automatic modification of the A and/or the B address; the N character is explained in greater detail under *Automatic Address Modification)*
4. B Address (three-character HSM address of the addend, subtrahend, destination location for an item, right boundary of a sector, etc.)

An instruction, then, is made up of eight RCA characters with the format OAAANBBB.

O	AAA	N	BBB
Operation Code	A address	N character	B address

In most of the instructions, the entire A address refers to a High-Speed Memory location (or tetrad) and the entire B address refers to another High-Speed Memory location (or tetrad). In these cases, the components of the A address (AAA) or the B address (BBB) need not be differentiated. In some instructions, however, only a portion of the A address or of the B address is used, or one component may designate one value and the other component another value; for example, one part of the A address may be used to specify a symbol and the other part to designate a count, or one part of the B address denotes the Tape Station and the other part is ignored. In these instructions, the components of the A address are referred to as A_1, A_2 and A_3, and the components of the B address as B_1, B_2 and B_3. This is illustrated below (under *Coded Instruction*).

Since each location in the High-Speed Memory is individually addressable, whenever instructions do not utilize the entire capacity of the A or B address, it is usually feasible for the programmer to employ the unused (ignored) portion of the address for the storage of constants.

Coded Instruction

Instructions are coded in octal notation. Since the octal equivalent of an RCA character consists of two octal digits, a coded instruction will contain sixteen octal digits.

Example of an instruction as coded by the programmer: 16 720003 10 400000

16	720003 (A_1 A_2 A_3)	10	400000 (B_1 B_2 B_3)
Operation Code	A address	N character	B address

Storage of Instructions

Instructions are stored sequentially in the High-Speed Memory. Each instruction is stored in two successive HSM tetrads (eight locations) so that the Operation Code falls in the leftmost (C_0) location of the first tetrad, with the A address in the remaining three locations (C_1, C_2 and C_3). The N character and the B address are in the same relative positions in the second tetrad—N in C_0 and B in C_1, C_2 and C_3.

Staticizing

An instruction can be interpreted and executed by Program Control only after it is brought out of the High-Speed Memory locations in which it has been stored, and its components placed in the proper registers. This process is called *staticizing* and is accomplished in two status levels.

A *status level* lasts 15 microseconds and is the term applied to a series of pulses which open certain paths over which information can travel. A status level that opens paths leading to or from the High-Speed Memory is called a *memory cycle. (Status level* and *memory cycle* are not synonymous, since not all status levels are concerned with opening the paths leading to or from the High-Speed Memory.) Each status level has a specific function. In staticizing of each instruction, the first status level brings the tetrad OAAA into the Memory Register, from which O is sent to the Normal Operation (NO) Register and AAA to the A Register. The second status level brings the tetrad NBBB into the Memory Register, from which N is sent to the N Register and BBB to the B Register.

Thus, staticizing time of 30 microseconds is constant for every instruction. The number of status levels involved, and their sequence, for execution (after staticizing) of a given instruction depends upon what must be accomplished by that instruction.

Automatic Address Modification

The first status level following staticizing checks the two octal digits comprising the N character in the N Register. If these digits are 00, the instruction will be executed as it was stored in the High-Speed Memory. If the first, or left-hand, octal digit is other than 0, the quantity stored in the location indexed by that digit will be added to the contents of the A Register (which has received the A address of the instruction) before the instruction is executed. If the second, or right-hand, octal digit is other than 0, the quantity stored in the location indexed by that digit will be added to the contents of the B Register (which has received the B address of the instruction) before the instruction is executed.

The addition is octal. In effect, subtraction may be performed by storing the eight's complement of the subtrahend (either the unmodified address or the contents of the indexed location). The N character can

thus effect a decrease or increase of the contents of either the A or the B Register, or of both.

Six status levels are used to modify each address. Automatic address modification time, therefore, is 90 microseconds if one address is modified, and 180 microseconds if both addresses are modified.

The locations, and their contents, accessed by the digits of the N character are called *Address Modifiers.* They permit modification not only of addresses, but data also.

The locations associated with each octal digit that could appear in the N character are as follows:

Octal Digit	*Location of Modifier*
1	HSM locations 000111 – 000113
2	HSM locations 000221 – 000223*
3	HSM locations 000131 – 000133
4	P Register†
5	HSM locations 000151 – 000153
6	T Register
7	HSM locations 000171 – 000173

* See STA, page 18.

† Note that the P Register always contains the address of the next instruction in sequence (not the address of the instruction currently being executed).

Four of these Address Modifiers (1, 3, 5 and 7) are *static;* that is, the contents remain as originally stored unless an instruction specifies that they be altered.

Three of the Address Modifiers (2, 4 and 6) are *dynamic.* The contents of the P Register will change with each instruction performed; the contents of the T Register will change with each instruction that utilizes this register; the contents of standard HSM locations 000221 – 000223 will change with each instruction in which STA is performed.

Example of Automatic Address Modification:

Assume the following instruction is stored in HSM locations 000460 – 000467 (000460 serves as the address of this instruction).

0	A	N	B
21	003100	03	000612

Assume, also, that a part of the memory looks like this:

Address	000131	000132	000133
Contents	04	20	13

The octal digit 3 in the N character of the instruction will access HSM locations 000131 – 000133 and cause the contents 042013 to be added (octally) to the contents of the B Register (000612). Since the left-hand octal digit of the N character is 0, the contents of the A Register (003100) will not be changed. Consequently, the instruction actually executed will be

21 003100 00 042625

The instruction remains in HSM locations 000460 – 000467 as it was originally written and stored:

21 003100 03 000612

PROGRAM CONTROL

Program Control is the arithmetic and control unit of the Computer. It interprets and executes the instructions stored in the High-Speed Memory, directs the sequence of operations in the Computer, controls operation of the on-line input-output devices, and performs the automatic accuracy checks. It includes the circuitry for electronic control, switching and buffering of up to eight input-output trunk lines.

Program Control includes a number of specialized (by function) devices. Those which are of interest to the programmer are diagrammed in Figure 5 and are briefly described below, along with certain automatic Program Control functions.

Registers

The *Memory Addressing Register* stores the HSM address of the tetrad to be processed. The capacity of this register is three RCA characters (six octal digits).

The *Memory Register* has a capacity of four RCA characters. It receives the tetrad contents that emerge from or are to be placed in the High-Speed Memory. A series of *Memory Output Gates* permit or inhibit entrance into the Memory Register of any or all of the four characters that emerge from the HSM.

The *P Register* holds the HSM address of the *next instruction* in sequence.

The *A Register* has a capacity of three RCA characters. It receives the A address of an instruction and, when necessary, holds the address of each character (or tetrad) being processed in the Normal Mode.

The *B Register* has a capacity of three RCA characters. It receives the B address of an instruction and, when necessary, holds the address of each character or tetrad being processed in the Normal Mode. When an instruction shifts from the Normal to the Simultaneous Mode, the B Register no longer holds the B address of the shifted instruction, but is utilized by the instruction which then occupies the Normal Mode.

The B address is not sent to the B Register in three of the forty-seven instructions: Transfer Control (71), Set Register (72) and Store Register (73).

The *S Register* has a capacity of three RCA characters. It assumes the function of the A Register for an operation when it shifts into the Simultaneous Mode.

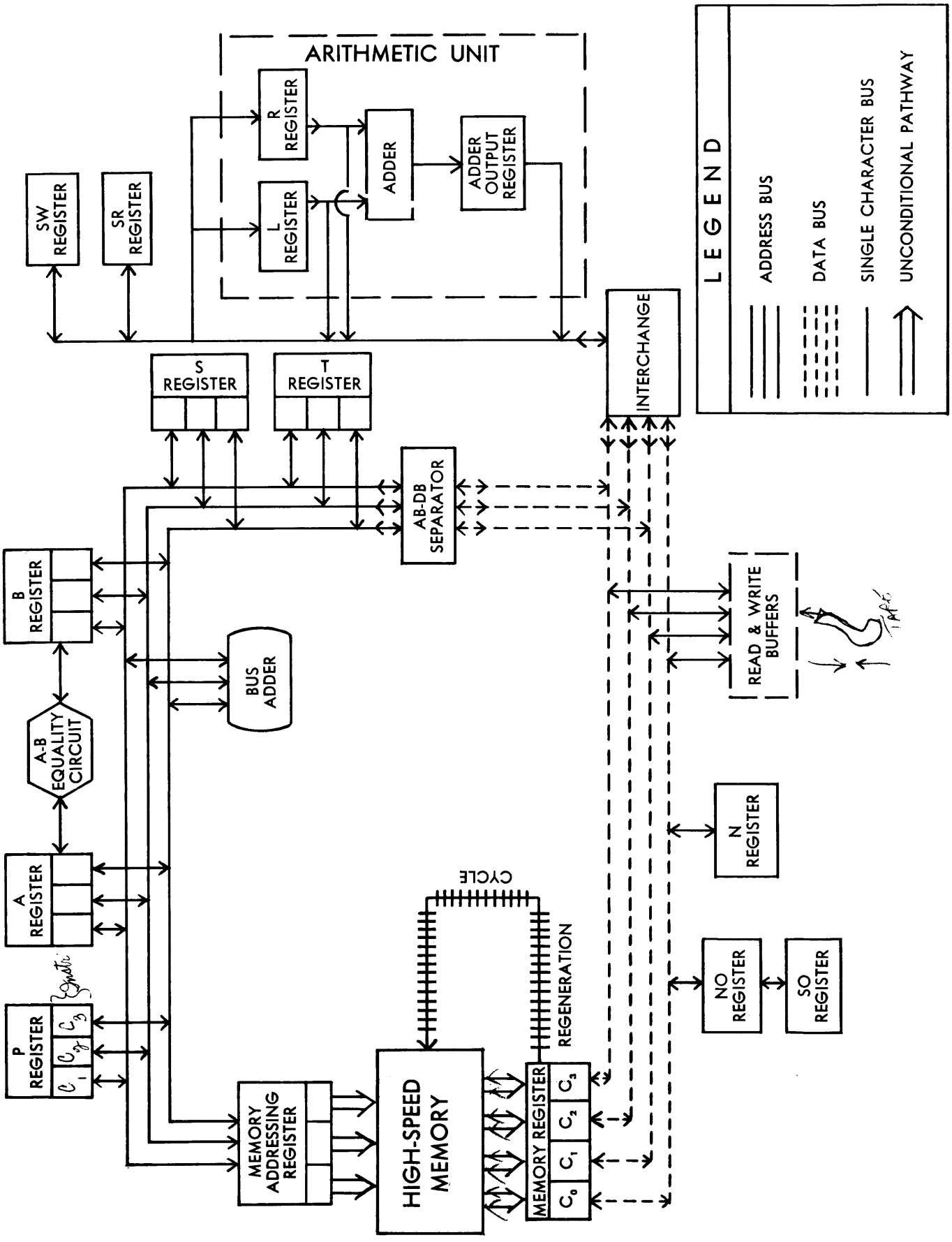


Figure 5—Simplified Diagram Showing Relationship of Buses and Registers in the Computer

The *T Register* has a capacity of three RCA characters. It holds the third address when required by an instruction (e.g., HSM address for the quotient in a Decimal Divide instruction). In some instructions it is used as an internal counter.

The *NO (Normal Operation) Register* has a capacity of one RCA character. It holds the Operation Code of the instruction currently being executed in the Normal Mode.

The *SO (Simultaneous Operation) Register* has a capacity of one RCA character. It holds the Operation Code of the instruction currently being executed in the Simultaneous Mode.

The *N Register* has a capacity of one RCA character. It holds the N Character of the currently processed instruction.

The *SR (Select Read) Register* has a capacity of one RCA character. It holds the address of the input device used in a read operation.

The *SW (Select Write) Register* has a capacity of one RCA character. It holds the address of the output device used in a write operation. [The SW Register is not used in the Print (On-Line Printer) instruction.]

Addressable Registers. A, B, P, S and T are all addressable registers. Their contents may be conveniently set and/or stored for subsequent program reference.

Buses

The *Address Bus* is a three-character pathway connecting the Memory Addressing Register with the A, B, P, S and T registers. Its terminal points are the Memory Addressing Register and the AB-DB Separator (see below, under Register Gates).

The *Data Bus* is a four-character pathway between the Memory Register and the Interchange (see below, under Register Gates), with branches to allow the C_1, C_2, and C_3 characters to enter the AB-DB Separator and thus pass into the Address Bus. The C_0 character is drawn from this bus for the NO or N Register.

The *Single-Character Bus* forms the connection between the Interchange and the Arithmetic Unit.

Register Gates

Basically, the Register Gates control the entrance into, and the exit from, the various registers and buses. In addition to the two sets at each register (one set for entrance and the other for exit of information), there are two major groups of gates controlling information flow between the various buses:

The *Address Bus — Data Bus (AB-DB) Separator* serves as the connection between the Address Bus and the Data Bus.

The *Interchange* links the Data Bus with the Single-Character Bus. It selects the character to be passed from the four-character Data Bus onto the Single-Character Bus. It also selects the Data Bus line that is to transport the one-character output of the Arithmetic Unit.

Bus Adder

The Bus Adder is located along the Address Bus, separated from it by a set of register gates. The function of the Bus Adder is to modify the contents of the various three-character address registers (A, B, P, S and T). The Bus Adder can increase the contents of these registers by $(01)_8$ or $(04)_8$ or can decrease them by $(01)_8$, $(04)_8$ or $(10)_8$, thus permitting register contents, for example, to be directly related to the currently processed characters or tetrads.

A-B Equality Circuit

The A-B Equality Circuit is located between the A and B registers and is used to compare their contents. When they are equal, the A Counter-B Counter Equality Flip-Flop is set, indicating to the Computer that the instruction-defined sector has been processed and the instruction can be terminated. In instructions that specify the left and right boundaries of the HSM sector to be processed, (1) the contents of the A Register are increased, toward A-B equality, when the direction of operation is from left to right and (2) the contents of the B Register are decreased, toward A-B equality, when the direction of operation is from right to left.

Arithmetic Unit

The Arithmetic Unit includes three registers, each of one-character capacity, and an Adder Circuit. In an arithmetic operation, the L (left) Register holds one character of one operand and the R (right) Register holds one character of the other operand. These characters are added and the sum is placed in the one-character Adder Output Register, with a flip-flop indicating whether a carry was generated. The sum is accuracy-checked in the Adder Output Register, then transferred onto the One-Character Bus to be returned, via the Memory Register, to the High-Speed Memory.

Previous Result Indicators (PRI's)

The Previous Result Indicators, a set of three flip-flops, preserve the sign of the result—or the zero result —of an arithmetic operation for automatic reference by a subsequent decision instruction. If the result is positive, the *Previous Result Positive (PRP)* flip-flop is set; if negative, the *Previous Result Negative (PRN)* flip-flop is set; if zero, the *Previous Result Zero (PRZ)* flip-flop is set.

The PRI's are set in all decimal arithmetic operations, in most of the binary arithmetic operations, in compare and certain search and transfer operations.

In addition to being automatically referenced by a decision instruction, the PRI's are directly addressable by program. Current PRI settings can be stored in instruction-designated HSM locations; also, any one of the PRI's can be set in accordance with program needs.

Automatic Storage of Final Contents of A Register (STA)

STA is an automatic operation which occurs at the conclusion of thirty-eight of the forty-seven RCA 501 instructions. In STA, the final contents of the A Register are automatically stored in the standard High-Speed Memory locations 000221 – 000223. This permits the indirect use of the A Register as a dynamic Address Modifier. It is also a highly convenient programming device which can be utilized to eliminate memory-searching time for subsequent processing. If STA were not automatically performed, the final A Register contents for one instruction would be destroyed as soon as the next instruction was staticized.

STA takes 15 microseconds and is performed in every instruction except those in the following list:

Transfer Control (71)
Conditional Transfer of Control (61)
Tally (66)
Set Register (72)
Store Register (73)
Tape Sense (63)
Sense Simultaneous Gate (65)
Control Simultaneous Gate (75)
Sense Simultaneous Mode (62)

Automatic Storage of Contents of P Register (STP)

STP is an automatic operation which occurs whenever control is to be transferred; that is, whenever the next instruction to be performed is not the one stored immediately following the current instruction. STP automatically stores the contents of the P Register in standard High-Speed Memory locations 000241 – 000243. Since the P Register always holds the address of the next instruction in sequence, the standard HSM locations may be accessed to construct, elsewhere in the HSM, a record of the instructions to which the program would have proceeded if transfer had not occurred.

STP always occurs in the following two instructions:

Transfer Control (71)
Sense Simultaneous Gate (65)

In the following four instructions, STP is performed *only when control is actually transferred:*

Conditional Transfer of Control (61)
Tally (66)
Tape Sense (63)
Sense Simultaneous Mode (62)

Unlike STA, the STP function is not a time factor; it does not add to basic instruction processing time.

Simultaneity (Time-Sharing Operations)

Simultaneity in the Computer is defined as coincident execution of two instructions, both or one of which is an input-output instruction.

All instructions are staticized in the Normal Mode. Some instructions must be totally executed in the Normal Mode. All but three of the input-output instructions can be completed in either mode and are termed *potentially simultaneous (PS)* instructions.

A potentially simultaneous instruction automatically shifts (any time after it is staticized) into, and is then completed in, Simultaneous if that mode is unoccupied by a previous instruction and if the Simultaneous Gate is open. This permits initiation (in the freed Normal Mode) of the next instruction in sequence. The two instructions are then executed with total or partial coincidence in time (time-shared). The exceptions are that two "read" instructions or two "write" instructions cannot be executed simultaneously. If, for instance, a "read" instruction is in process in the Simultaneous Mode and another "read" instruction is staticized in the Normal Mode, the latter instruction is not executed until the instruction in the Simultaneous Mode has been completed.

Reading and writing may be accomplished simultaneously as long as the two instructions do not involve the same Tape Station. If, for example, a "read" from Tape Station 20 is staticized while writing is in process at that Tape Station, the "read" will not be executed until writing has been completed.

Trunk $(77)_8$, however, can be time-shared by a paper tape "read" and a "write-out" to the Monitor Printer.

Time-shared electronics in the Computer permits the following simultaneous operations:

compute and magnetic tape write
compute and paper tape punch
compute and monitor print

paper advance* and compute
paper advance* and magnetic tape write
paper advance* and paper tape punch
paper advance* and monitor print

paper tape read and compute
paper tape read and magnetic tape write
paper tape read and paper advance*
paper tape read and paper tape punch
paper tape read and monitor print

* With an On-Line Printer.

magnetic tape read and compute
magnetic tape read and magnetic tape write
magnetic tape read and paper advance*
magnetic tape read and paper tape punch
magnetic tape read and monitor print

Any one of the above-listed combinations is possible while, at the same time, any number of tapes are rewinding.

Simultaneous operation within the Computer is made possible by the low-duty cycle of the High-Speed Memory during execution of most of the input-output instructions. The majority of the time required for the execution of a tape instruction, for example, is used up in tape movement; the High-Speed Memory is involved only a very small fraction of that time. If properly controlled, therefore, the memory is available for other functions while it is waiting for the tape to be advanced. In order to accomplish this with a minimum of buffering and additional hardware, an interruption technique is employed. That is, the sequence of instructions being executed simultaneously with the tape function is automatically interrupted when the memory must receive or transmit information in connection with the tape operation.

The RCA 501 includes two Read buffers and two Write buffers; each has a capacity of four characters. One character from tape is clocked into the first Read Buffer in 30 microseconds, and the buffer is filled in 120 microseconds. The entire contents shift into the second Read Buffer and then are transferred in parallel, in one status level (15 microseconds) into an HSM tetrad. The 30-microsecond clocking of characters from tape to buffer continues uninterrupted until the read instruction has been completed.

* With an On-Line Printer.

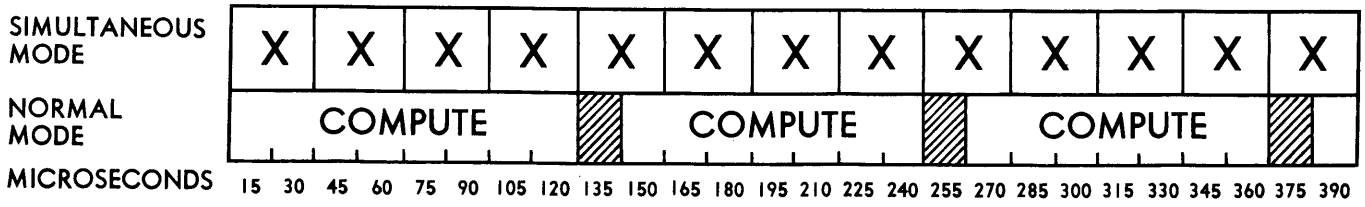
During 105 μs of the 120-μs buffer-filling time, the Computer is free to execute another instruction or instructions. The number of instructions that can be executed simultaneously with any one read instruction depends upon the number of characters to be read in. Execution of these instructions is interrupted only for the 15-microsecond, transfer time.

These same factors apply to "write" operations, except that the 15-microsecond interruption occurs with tetrad transfer from the HSM to the Write Buffer and the Computer is free during seven-eighths (105 μs) of the 120 microseconds required to write out the four characters.

Assuming a "read" instruction being executed in the Simultaneous Mode and "compute" functions (add, locate, compare, etc.) in the Normal Mode, simultaneity may be illustrated by the diagram below. (Each x represents a character read in from tape; the shaded areas represent interruption of "compute" during buffer-to-HSM transfer.)

The programmer can control the use of simultaneity by employing instructions which sense the state of and control the Simultaneous Gate. When open, this gate permits transition of potentially simultaneous instructions from the Normal to the Simultaneous Mode. When closed, the gate prevents such transition so that all instructions are performed in the Normal Mode. It is thus possible to run programs entirely in the Normal Mode or to bracket off certain parts of programs in which transition to the Simultaneous Mode is not desirable.

In addition, use of the instruction "Sense Simultaneous Mode" provides the information that the Simultaneous Mode is engaged in a "read," a "write" or a paper advance, or that it is totally unengaged.



THE RCA 501 INSTRUCTIONS

INTRODUCTION

The Computer operates under the direction of forty-seven basic, wired-in, two-address instructions. For descriptive purposes, these instructions may be classified into four general categories: (1) Input-Output, (2) Data-Handling, (3) Arithmetic and (4) Decision and Control.

Input-Output Instructions

These instructions enable the Computer to communicate with the on-line peripheral devices (Magnetic Tape Stations, Paper Tape Reader, Monitor Printer and Paper Tape Punch, On-Line Printer). They perform the functions of positioning or searching tapes, bringing data from an input medium into the Computer, or sending data from the Computer to an output medium.

Most of the twelve instructions in this group are potentially simultaneous (PS); i.e., they can be executed in the Simultaneous Mode, so that operational time for these instructions can overlap that of other instructions. (See Simultaneity, page 18.) Two instructions, Single Sector Write (11) and Multiple Sector Write (13), are not PS instructions. One instruction, Print (02), occupies both modes and cannot be executed simultaneously with any other Computer-controlled operation.

One input-output instruction, Rewind to BTC (17), is initiated by the Computer but, once underway, operates completely independent of the Computer. Any number of tapes may be rewinding while two instructions (unrelated to the rewinding tapes) are being simultaneously executed.

After staticizing and address modification, a "read" instruction is automatically stored in standard High-Speed Memory locations 000020 – 000027. The instruction in the standard locations is then available for a Rollback (rerun) routine if an error is detected by the checking circuitry (see description of Rollback, Appendix I) or for other program needs.

Three "write" instructions — Linear Write (12), Single Sector Write (11) and Multiple Sector Write (13)—are automatically stored (after staticizing and address modification) in standard High-Speed Memory locations 000030 – 000037.

Data-Handling Instructions

These are non-arithmetic instructions for manipulation of data stored in the High-Speed Memory. The thirteen instructions included in this group permit operation control by symbol or by address, and transfer of data with or without editing.

With the exception of Sector Locate *n*th Symbol (31), Zero Suppress (32) and Random Distribute (27), all data-handling sector and item instructions operate from right to left, i.e., processing begins with the larger specified HSM address and progresses to the smaller or to the Item Separator symbol.

Arithmetic Instructions

Four of the eleven instructions in this group are decimal arithmetic instructions, five are binary, and two are used to alter the bit configuration of an operand.

The decimal instructions operate in accordance with arithmetic rules and are designed to handle operands of unequal and practically unlimited length. They employ the Computer's ability to recognize control symbols, so that the arithmetic process, in effect, is performed with alignment of the least significant digits and is terminated when the Item Separator symbol (ISS) or a space character to the left of the longer operand is encountered. Thus, the need for the programmer to pre-position operands, by shifting, is largely eliminated, since proper alignment will be achieved even if one (or both) of the operands as addressed contains a series of spaces to the right of the sign.

The binary instructions handle operands of equal but unlimited length. Here length is not defined by the presence of a control symbol, but by address specification. Alignment of the operands must be programmed, since it is not automatically performed as in the decimal instructions. All the characters in the operands are treated as numerics, and the bits in each bit position are added together in accordance with the stated binary rules.

Two instructions, Logical "and" (47) and Logical "or" (46), constitute what may be considered a separate arithmetic category. They are used to alter the bit configuration of an operand, by the employment of a second operand to "mask out" or to insert "1" bits.

The Previous Result Indicators (PRI's) preserve the sign of the result of an arithmetic instruction for reference by a subsequent decision instruction. (See page 17.)

Decision and Control Instructions

Seven of the eleven decision and control instructions influence the sequence of operation. Four of these are conditional; that is, they choose a path according to PRI settings, the kind of instruction currently in the Simultaneous Mode, the state of the Simultaneous Gates, or the status of a designated Tape Station. Two of the seven are unconditional commands, and one enables the Computer to execute the same subroutine any designated number of times.

Two instructions enable the programmer to address registers directly; one instruction controls the Simultaneous Gates; and one stops Computer operation.

Logic

A *logic* subsection supplements the general description prefacing each instruction. Internal logic is described not in every detail, but only to the extent that it contributes to attainment of the objectives of a programmers' reference manual: (1) to help the programmer gain a better understanding of Computer operation, (2) to permit him to vary the instructions, and their application, for individual problem solution, and (3) to enable him to develop advanced programming techniques.

Timing

Owing to the variable item length concept in the RCA 501 System, instruction times can be expressed only as a function of the number of significant characters involved in a given operation. For example, the time required to write out to tape depends on the number of characters to be written, and the time required to add two numbers together depends on the number of digits in the operands.

The timing formulas for read and write instructions are applicable to magnetic tape since this is the most frequently used input-output medium. For other types of input or output, timing is governed largely by the input or output rate of the device used (see page 4 for input-output performance, Monitor Printer, Paper Tape Punch, Paper Tape Reader).

The time, or timing formula, listed for each instruction does not include staticizing, automatic address modification, or STA time. As stated previously:

Staticizing time is 30 microseconds and is constant for each instruction.

Automatic address modification occupies 90 microseconds if the A or the B address is modified; 180 microseconds if both A and B are modified.

STA occupies 15 microseconds. It is automatically performed in every instruction except in those in which it is specifically stated otherwise.

Examples

Wherever possible, the examples that accompany the instructions include a representation of the affected portion or portions of the High-Speed Memory. The contents of the HSM locations are shown as *octal values* in most of the binary instructions, and as decimal digits, alphabetic characters, symbols, etc., in other instructions.

In each example, the subhead *HSM before Instruction is executed* is to be interpreted as *before execution* but *after staticizing.* The contents of the memory locations are not affected by staticizing. The initial register settings $[(A)_i, (B)_i, (T)_i]$, however, reflect register contents after staticizing.

Order of Presentation

The instructions that follow are arranged in logical order of presentation, as follows:

72: Set Register (SET)

General Description

This instruction replaces the contents of a specified register with the A address of the instruction or sets a PRI.

Format

A address — contains the ~~actual value (not the HSM~~ location thereof) to be placed in the register designated by the B_1 character. The A address for setting the PRI's is as follows:

$(000001)_8$ sets PRN
$(000002)_8$ sets PRZ
$(000004)_8$ sets PRP

B address:

B_1 — specifies the register (or PRI's) to be set:

B_1 Character	*Register Selected*
$(10)_8$	PRI's
$(20)_8$	A Register
$(40)_8$	P Register
$(60)_8$	T Register

B_2B_3 — ignored.

Outline of Logic

The B_1 character is examined in the SW or the SR Register (the B address is not sent to the B Register) and used to select the register whose contents are to be replaced. The contents of the A address are then placed into the selected register. If the B_1 character is anything other than one of the values listed above, the instruction will not be executed, but the timing will be the same as if it had been executed. No alarm stop will occur; the program will continue to the next instruction in sequence. If the A Register is designated to be set, the instruction goes through STA; otherwise, it does not.

Addressable Registers Used

A
Register specified

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$ (contents remaining from previous instruction)

Timing

15 μs, unless the A Register is to be set, in which case time is 30 μs.

73: Store Register (STR)

General Description

This instruction places the contents of a selected register (or PRI setting) into the rightmost three locations of a designated tetrad.

Format

A address — specifies the tetrad that is to receive the contents of the designated register.

B address:

B_1 — specifies the PRI's or the register whose contents are to be stored.

B_1 Character	*Register Selected*
$(10)_8$	PRI's
$(30)_8$	B Register
$(40)_8$	P Register
$(50)_8$	S Register
$(60)_8$	T Register

B_2B_3 — ignored.

Outline of Logic

The B_1 character is examined in the SW or the SR Register (the B address is not sent to the B Register). The contents of the designated register are then stored in C_1, C_2 and C_3 of the tetrad specified by the A address; the original C_0 remains undisturbed.

If the B_1 character in an STR instruction is $(10)_8$, the value that will be stored in the tetrad specified by the A address will be $(000001)_8$ if PRN is set, $(000002)_8$ if PRZ is set, and $(000004)_8$ if PRP is set.

~~If the B_1 character is not one of the values listed (Format), the instruction will not be executed, but the timing will be the same as if it had been executed. No alarm stop will occur~~; the program will continue to the next instruction in sequence. (Note that the A Register is not included in the list.) This instruction does not go through STA.

Addressable Registers Used

A Register specified

Final Register Contents

$(A)_f = (A)_i - (01)_8$
$(B)_f = (B)_i$ (contents remaining from previous instruction)

Timing

15 μs.

Example (STR)

Instruction: 73 010320 00 100000
Assumption: PRP set as a result of a prior operation.

HSM before Instruction is executed:

	14	15	16	17	20	21	22	23	24	25	26	27
0103	01	16	11	30	72	00	00	00	00	10	00	00

$(A)_i$

HSM after Instruction:

	14	15	16	17	20	21	22	23	24	25	26	27
0103	01	16	11	30	72	00	00	04	00	10	00	00

$(A)_f$

14: Linear Read Forward (LRF)

General Description

This instruction brings one full message from magnetic or punched paper tape into the HSM. It is a potentially simultaneous instruction.

Format

A address — specifies the tetrad which is to receive the SM, ED or EF. The SM (ED or EF) will be placed in C_0 of the tetrad no matter which of the four locations is specified by the A address.

B address:

B_1 — address of the input unit; [$(77)_8$ addresses the Paper Tape Reader].

B_2B_3 — ignored.

Direction of Operation

Left to right.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the designated input device. Coincident with each pulse (or sprocket hole) in the timing track, a character is brought into a recognition circuit. A check is performed to verify that the first character is an SM, ED or EF. The SM and the three characters following it are admitted into the Read Buffer, and then placed (four-character parallel transfer) in the HSM tetrad specified by the contents of the A Register. Tape-to-Buffer transfer continues to overlap Buffer-to-HSM transfer until an EM is sensed, at which point a stop signal is sent to the input device.

If the EM is not placed in C_3 of a tetrad, then one, two or three spaces are generated to fill out the tetrad. The A (or S) Register is reset so that it holds the HSM address of the EM.

The control symbols ED and EF, when standing alone, are treated as complete messages in themselves. Gaps on tape intervening between the SM and EM are ignored by this instruction.

Addressable Registers Used

A or S B

Final Register Contents

$(A)_f$ or $(S)_f$ = the HSM address of the location containing the EM.

$(B)_f = (B)_i$ (unless the instruction is concluded in the Simultaneous Mode).

Timing

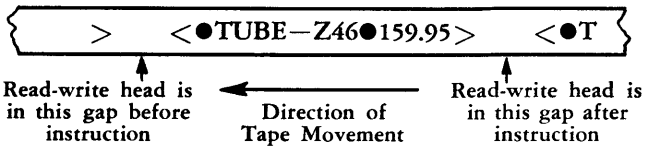
Total time in milliseconds = $3.575 + .03n$

where n is the total number of characters transferred.

Example (LRF)

Instruction: 14 116726 00 060000

Tape (on Tape Station 06):



HSM before Instruction is executed:

1167	23	24	25	26	27	30	31	32	33	34	35	36	37
	A	B	C	D	E	F	G	H	I	J	K	L	M

$(A)_i$

1167	40	41	42	43	44	45	46	47	50	51	52	53	54
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

HSM after Instruction:

1167	23	24	25	26	27	30	31	32	33	34	35	36	37
	A	<	●	T	U	B	E	−	Z	4	6	●	1

1167	40	41	42	43	44	45	46	47	50	51	52	53	54
	5	9	.	9	5	>	−	−	V	W	X	Y	Z

$(A)_f$

Final register contents:

$(A)_f$ or $(S)_f$ = 116745

$(B)_f$ = 060000 (unless the instruction is concluded in the Simultaneous Mode).

Time:

3.575 + (.03 x 18) = 4.115 ms.

04: Linear Read Reverse (LRR)

General Description

This instruction transfers one message from magnetic or paper tape to the HSM. It is a potentially simultaneous instruction. Though the tape is read in reverse, the characters will be placed in the HSM in their proper relative positions. LRR is most useful in sort routines since it saves rewind time.

Format

A address — specifies the tetrad in which the EM is to be stored. Any one of the four locations may be specified; the EM, however, will always be placed in C_3.

B address:

B_1 — specifies the input device. [$(77)_8$ addresses the Paper Tape Reader.]

B_2B_3 — ignored.

Direction of Operation

Right to left.

Direction of Tape Movement

Magnetic tape moves in reverse. Paper tape moves in the forward direction. In LRR, however, characters on paper tape must come in as if the tape were moving in reverse, i.e., EM first and SM last. This may be accomplished by manually reversing and inverting a paper tape which was prepared in normal fashion. (LRR from paper tape is used in program testing.)

Outline of Logic

A start signal is sent to the designated input device. Coincident with each pulse (or sprocket hole) in the timing track, a character is brought into a recognition circuit. A check is performed to verify that the first character is an EM, ED or EF. (If the first character is not one of these three, an alarm stop occurs.) The EM and the three characters following it are admitted into the Read Buffer and then placed (tetrad transfer) in the HSM. Characters are placed in the HSM beginning with the rightmost location of the tetrad specified by the contents of the A Register. The operation stops when an SM (ED or EF) is sensed.

If the SM (ED or EF) is placed in C_1, C_2 or C_3 of a tetrad, one, two or three spaces, respectively, will be generated to fill out the tetrad. In any event, the A (or S) Register is reset so that it holds the HSM address of the SM (ED or EF).

Gaps on tape intervening between the EM and SM are ignored by this instruction; the message is read as if it were free of gaps.

Addressable Registers Used

A (or S if the operation is concluded in the Simultaneous Mode)

B

Final Register Contents

$(A)_f$ or $(S)_f$ = address of HSM location of SM, ED or EF.

$(B)_f = (B)_i$ (unless instruction is concluded in Simultaneous Mode).

Timing

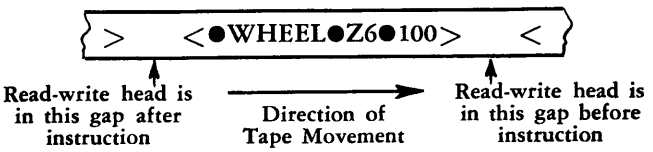
Total time in milliseconds = $3.575 + .03n$

where n is the total number of characters transferred.

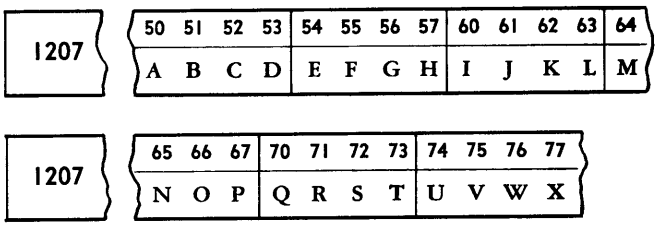
Example (LRR)

Instruction: 04 120771 00 130000

Tape (on Tape Station 13):

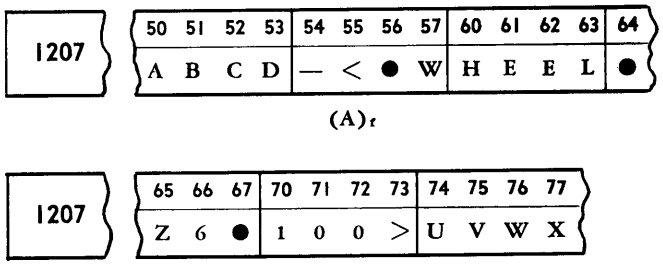


HSM before Instruction is executed:



$(A)_i$

HSM after Instruction:



$(A)_f$

Final register contents:

$(A)_f$ or $(S)_f = 120755$

$(B)_f = 130000$ (unless instruction is concluded in the Simultaneous Mode).

Time:

$3.575 + (.03 \times 15) = 4.025$ ms.

15: Block Read Forward (BRF)

General Description

This instruction brings a block of characters from magnetic or punched paper tape into the HSM. Transfer from tape begins with the first character following a gap, and ends when the next gap is sensed. This instruction is potentially simultaneous.

(See discussion of automatic decoding of characters on block read from paper tape, page 10.)

Format

A address — specifies the tetrad which will receive the first character in the block; this character will always be placed in C_0 of the tetrad no matter which of its four locations is specified.

B address:

B_1 — address of the input unit; $[(77)_8$ addresses the Paper Tape Reader].

B_2B_3 — ignored.

Direction of Operation

Left to right.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the designated input unit. When the first four characters following the gap have been placed in the Buffer, they are transferred (four-character parallel transfer) into the HSM. Tape-to-Buffer transfer continues to overlap Buffer-to-HSM transfer until a gap is detected by the input device. If the last character in the block is not placed in C_3 of a tetrad, one, two or three spaces are generated to fill out the tetrad. The A (or S) Register is reset so that it holds the HSM address of the last character read.

Addressable Registers Used

A or S B

Final Register Contents

$(A)_f$ or $(S)_f$ = HSM location containing the last character read.

$(B)_f = (B)_i$ (unless the operation is concluded in the Simultaneous Mode).

Timing

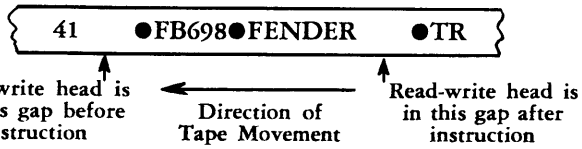
Total time in milliseconds = $3.575 + .03\ (n + 6)$

where n is the total number of characters transferred.

Example (BRF)

Instruction: 15 123055 00 010000

Tape (on Tape Station 01):



HSM before Instruction is executed:

1230	50	51	52	53	54	55	56	57	60	61	62	63	64
	A	B	C	D	E	F	G	H	I	J	K	L	M

$(A)_i$

1230	65	66	67	70	71	72	73	74	75	76	77
	N	O	P	Q	R	S	T	U	V	W	X

HSM after Instruction:

1230	50	51	52	53	54	55	56	57	60	61	62	63	64
	A	B	C	D	●	F	B	6	9	8	●	F	E

1230	65	66	67	70	71	72	73	74	75	76	77
	N	D	E	R	–	–	–	U	V	W	X

$(A)_f$

Final register contents:

$(A)_f$ or $(S)_f$ = 123070

$(B)_f$ = 010000 (unless the operation is concluded in the Simultaneous Mode).

Time:

$3.575 + .03\ (13 + 6) = 4.145$ ms.

05: Block Read Reverse (BRR)

General Description

This instruction transfers a block of characters from magnetic or punched paper tape into the HSM. Transfer begins with the first character following a gap and ends when the next gap is sensed. Though the tape moves in reverse, the characters will be placed in the HSM in their proper relative positions. The instruction is potentially simultaneous.

(See discussion of automatic decoding of characters on block read from paper tape, page 10.)

Format

A address — specifies the tetrad which is to receive the first character (following a gap) read from the tape. Any one of the four locations in the tetrad may be specified; the first character, however, will always be placed in C_3.

B address:

B_1 — specifies the input device. $[(77)_8$ addresses the Paper Tape Reader.]

B_2B_3 — ignored.

Direction of Operation

Right to left.

Direction of Tape Movement

Magnetic tape moves in reverse. Paper tape is manually reversed and inverted and moves in the forward direction.

Outline of Logic

A start signal is sent to the designated input unit. The first four characters following the gap are placed in the Read Buffer and then (four-character parallel transfer) into the HSM tetrad specified by the A Register. The first character read from tape is placed in C_3. The operation continues until a gap is detected. If the last character of the block is placed in C_1, C_2 or C_3 of a tetrad, one, two or three spaces, respectively, will be generated to fill out the tetrad. The A (or S) Register is reset so that it holds the HSM address of the last character read.

Addressable Registers Used

A or S B

Final Register Contents

$(A)_f$ or $(S)_f$ = HSM location containing the last character read.

$(B)_f = (B)_i$ (unless the operation is concluded in the Simultaneous Mode).

Timing

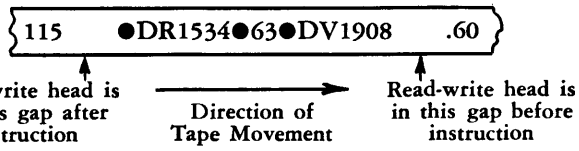
Total time in milliseconds = $3.575 + .03\ (n + 6)$

where n is the total number of characters transferred.

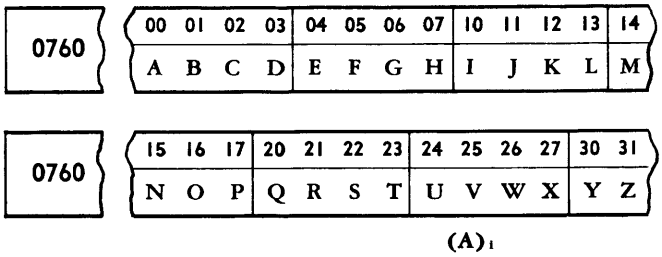
Example (BRR)

Instruction: 05 076025 00 200000

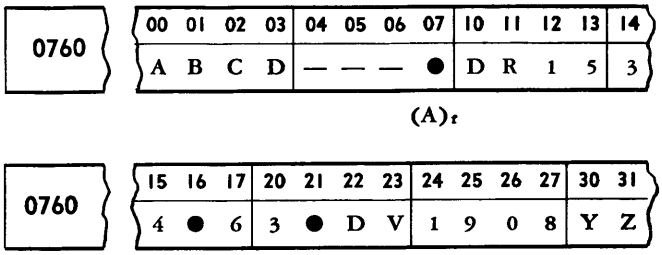
Tape (on Tape Station 20):



HSM before Instruction is executed:



HSM after Instruction:



Final register contents:

$(A)_f$ or $(S)_f = 076007$

$(B)_f = 200000$ (unless the operation is concluded in the Simultaneous Mode).

Time:

$3.575 + .03\ (17 + 6) = 4.265$ ms.

12: Linear Write (LW)

General Description

This instruction transfers one message from the HSM to magnetic tape, the Monitor Printer or to paper tape via the Monitor Printer. It is a potentially simultaneous instruction.

Format

A address — HSM location of the leftmost character in the message to be written.

B address:

B_1 — address of the output unit; [$(77)_8$ specifies the Monitor Printer, or the Paper Tape Punch via the Monitor Printer.]

B_2B_3 — ignored.

Direction of Operation

Left to right.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the output unit. If this is a Tape Station, transfer of characters is delayed until the tape is moving at the proper speed. If output is to or via the Monitor Printer, there is no delay.

The tetrad containing the leftmost character of the message, specified by the A Register, is placed in the Memory Register. If this character is EM, EF or ED, it is written out and the operation stops. If it is not EM, EF or ED, it is sent to the Write Buffer along with the adjacent characters (to the right) in the tetrad. HSM-to-Buffer transfer is overlapped with Buffer-to-Tape (or Monitor Printer) transfer. The operation is terminated when an EM is written out.

Addressable Registers Used

A or S B

Final Register Contents

$(A)_f$ or $(S)_f$ = HSM address of EM, EF or ED

$(B)_f = (B)_i$ (unless the instruction is concluded in the Simultaneous Mode).

Timing

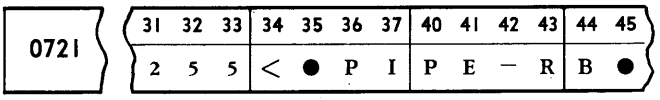
Total time in milliseconds = $3.575 + .03n$

where n is the total number of characters transferred.

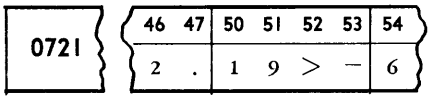
Example (LW)

Instruction: 12 072134 00 020000

HSM before and after Instruction is executed:

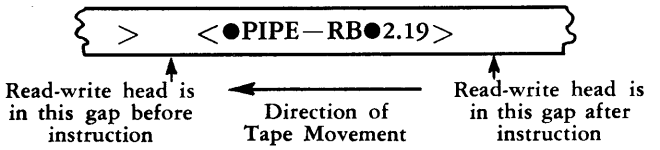


$(A)_i$



$(A)_f$

Tape (on Tape Station 02) contents after execution:



Final register contents:

$(A)_f$ or $(S)_f$ = 072152

$(B)_f$ = 020000 (unless the instruction is concluded in the Simultaneous Mode).

Time:

3.575 + (.03 x 15) = 4.025 ms.

11: Single Sector Write (SSW)

General Description

This instruction writes onto magnetic tape (or the Monitor Printer), in block format, the contents of a sector of any size, located in any area of the HSM. This instruction can be executed only in the Normal Mode. However, a prior "read" may, at the same time, be in the Simultaneous Mode.

Format

A address — leftmost HSM location of sector.

B address — rightmost HSM location of sector.

Preset T Register [see Set Register Instruction (72), page 23].

T_1 — address of output unit. [(77_8) will select the Monitor Printer or the Paper Tape Punch via the Monitor Printer.]

T_2T_3 — ignored.

Direction of Operation

Left to right.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the designated output unit. The contents of the tetrad addressed by the A Register are read out of the HSM and into the Write Buffer. The contents of the A Register are increased by $(04)_8$. Beginning with the character from the location specified by the A address and proceeding to the right, the characters are written onto tape (or the Monitor Printer) while the next four characters are being read into the Buffer. HSM-to-Buffer transfer continues to overlap Buffer-to-Tape transfer until the last tetrad in the sector has been transferred to the Buffer. The appropriate number of characters from the last tetrad are written out and the A Register is reset so that it holds the HSM address of the last character written out. The operation is then terminated.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (B)_i$ $(B)_f = (B)_i$ $(T)_f = (T)_i$

Timing

Total time in milliseconds = $3.575 + .03n$

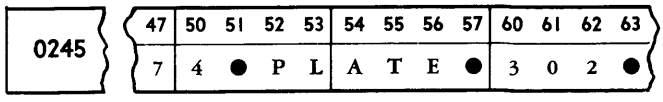
where n is the total number of characters transferred.

Example (SSW)

Instruction: 11 024551 00 024562

T Register is previously set to 060000

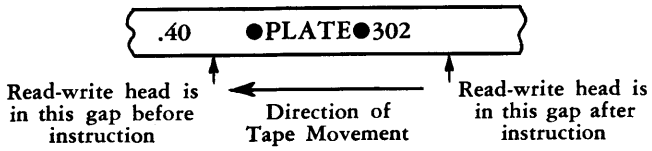
HSM before and after Instruction:



$(A)_i$

$(B)_i$
$(B)_f$
$(A)_f$

Tape (on Tape Station 06) after execution:



Final register contents:

$(A)_f = 024562$ $(B)_f = 024562$ $(T)_f = 060000$

Time:

3.575 + (.03 x 10) = 3.875 ms.

13: Multiple Sector Write (MSW)

General Description

This instruction writes onto magnetic tape (or the Monitor Printer), a single block comprised of the contents of any number of sectors taken from various parts of the HSM under the direction of a stored list of addresses. This instruction is executed only in the Normal Mode.

Format

A address — specifies the leftmost tetrad of the list of addresses. The contents of each tetrad in the list must be in the form KXXX, where XXX is the HSM address of the leftmost character of the sector to be written and K is the number (octal count) of character locations in the sector. When K is initially $(00)_8$, sixty-four characters are written out.

In the last address in the list, XXX must be $(000000)_8$ and K is ignored.

B address:

B_1 — address of the output unit; [$(77)_8$ addresses the Monitor Printer or the Paper Tape Punch via the Monitor Printer].

B_2B_3 — ignored.

Direction of Operation

Left to right.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the designated output device. The XXX of the first address is placed in the T Register and in the Bus Adder, and K is placed in the L Register. XXX is examined first. If it is $(000000)_8$ (tested in the Bus Adder), the operation ends. If XXX is not $(000000)_8$, characters are read out of the HSM, by tetrad, and into the Write Buffer. As each tetrad is sent to the Buffer, the quantity stored in the L Register is decreased by $(04)_8$, and the result is tested. The first Buffer-to-Tape transfer will be less than four characters if XXX does not refer to C_0 of a tetrad. Depending upon K, the last transfer may also be less than four. The number to be subtracted from the contents of the L Register is automatically adjusted in accordance with the initial K and XXX, so that final K (in the L Register) will always be $(00)_8$. When the result is $(00)_8$, the next address in the list is brought out of the HSM and placed in the T Register and in the Bus Adder.

In MSW, blanks generated on magnetic tape between one KXXX and the next will constitute less than 75 microseconds of tape time.

The contents of the A Register are increased by $(04)_8$ as each tetrad in the list is processed. At the conclusion of MSW, the A Register holds the address of the terminal tetrad.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f$ = HSM tetrad containing terminal address in list [XXX = $(000000)_8$]

$(B)_f = (B)_i$

$(T)_f = (000000)_8$

Timing

Total time in milliseconds = $3.575 + .03n + .03\ (m - 1)$

where n is the total number of characters transferred, m is the total number of addresses in the list.

Example (MSW)

Instruction: 13 067000 00 060000

HSM before Instruction is executed:

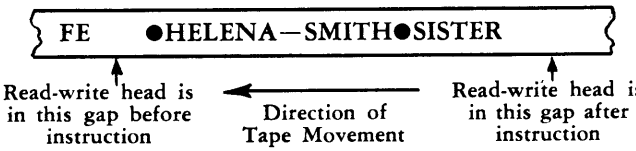
	22	23	24	25	26	27	30	31	32	33	34	35	36
0417	4	●	H	E	L	E	N	A	–	S	M	I	T

	37	40	41	42	43	44	45	46	47	50	51	52	53
0417	H	●	1	7	8	5	–	C	O	L	L	I	N

	56	57	60	61	62	63	64	65	66	67	70
0420	–	S	T	●	S	I	S	T	E	R	●

	00	01	02	03	04	05	06	07	10	11	12	13	14
0670	15	04	17	23	07	04	20	61	56	00	00	00	26
	$(A)_i$				$(A)_f$								

Tape (on Tape Station 06):



Final register contents:

$(A)_f = 067010$ $(B)_f = 060000$ $(T)_f = 000000$

Time:

3.575 + (.03 x 20) + (.03 x 2) = 4.235 ms.

06: Unwind *n* Symbols (UNS)

General Description

This instruction causes a selected magnetic tape to be moved forward until a specified number of a designated symbol have been counted. The HSM is not altered. The instruction is potentially simultaneous, but no "read" instruction can be executed simultaneously with it.

Format

A address:

A_1 — designates the symbol; this may be EM, ED, EF or Gap. Gap is designated by $(00)_8$.

A_2A_3 — specifies (in octal count) the number of symbols to be counted; this may be $(0000)_8$ to $(7777)_8$.

B address:

B_1 — designates the address of the Tape Station.

B_2B_3 — ignored.

Direction of Tape Movement

Forward.

Outline of Logic

A start signal is sent to the designated Tape Station. The characters on the tape pass into the recognition circuit, but do not enter the Read Buffer. When an instance of the specified symbol is found, the Computer is signaled and $(0001)_8$ is subtracted from A_2A_3 by the Bus Adder. When the output of the Bus Adder is $(0000)_8$, a stop signal is sent to the input device and the operation is concluded.

If the PET is sensed, the Tape Station and the Computer will stop.

If *n* is initially $(0000)_8$, the tape will not move. The instruction, however, will be stored in the standard HSM locations in which a read instruction is stored.

When unwinding EM's, an alarm stop occurs if the data are not in message format.

When unwinding gaps, a CIG alarm stop occurs if there are fewer than eight characters in a message or block. Exception: ED and EF stand alone (they may never appear within a message).

An alarm stop occurs if a nonpermissible symbol is specified.

Addressable Registers Used

A or S	B

Final Register Contents

If the operation is concluded in the Normal Mode:

$(A_1)_f = (A_1)_i$

$(A_2A_3)_f = (0000)_8$ unless the PET is reached (alarm), in which case $(A_2A_3)_f = (A_2A_3)_i$ minus the number of symbols counted.

$(B)_f = (B)_i$

If the operation is concluded in the Simultaneous Mode:

$(S_1)_f = (A_1)_i$

$(S_2S_3)_f = (0000)_8$ unless the PET is reached, in which case $(S_2S_3)_f = (A_2A_3)_i$ minus the number of symbols counted.

Timing

Total time in milliseconds $= 3.575 + .03n + 4m$

where *n* is the total number of characters read, including symbols counted and *m* is the number of gaps encountered.

Example (UNS)

Instruction: 06 750015 00 200000

This instruction will move the tape (on Tape Station 20) forward through 13 End Message symbols, unless the PET is reached before the full count. With the count completed, the tape stops with the Read-Write head in the gap following the thirteenth EM, positioned, for example, for a Linear Read Forward of the subsequent message, or a Linear Read Reverse of the message associated with the thirteenth EM counted.

16: Rewind *n* Symbols (RNS)

General Description

This instruction causes a selected magnetic tape to be moved backward through a specified number of symbols. This instruction is potentially simultaneous, but no "read" instruction can be executed simultaneously with it. The HSM is not altered.

Format

A address:

A_1 — designates the symbol, which may be SM, EF, ED or Gap.

A_2A_3 — specifies (in octal count) the number of symbols to be counted.

B address:

B_1 — designates the address of the Tape Station.

B_2B_3 — ignored.

Direction of Tape Movement

Reverse.

Outline of Logic

A start signal is sent to the designated Tape Station. The characters on the tape pass into the recognition circuit, but do not enter the Read Buffer. When an instance of the specified symbol is found, the Computer is signaled and $(0001)_8$ is subtracted from A_2A_3 by the Bus Adder. When the output of the Bus Adder is $(0000)_8$, a stop signal is sent to the input device and the operation is concluded.

If $(A_2A_3)_i = (0000)_8$, the tape does not move. The instruction, however, is stored in the standard HSM locations for a read instruction.

If $n = (0000)_8$ initially, the tape does not move. The instruction, however, is stored in the standard HSM locations for a read instruction.

When rewinding SM's, an alarm stop occurs if the data are not in message format.

When rewinding gaps, if there are fewer than eight characters in a message or block, a CIG alarm stop will occur. Exception: ED and EF stand alone (they may never appear within a message).

If a nonpermissible symbol is specified, an alarm stop will occur.

Addressable Registers Used

A or S B

Final Register Contents

If the operation is concluded in the Normal Mode:

$(A_1)_f = (A_1)_i$

$(A_2A_3)_f = (0000)_8$ unless the BTC was reached, in which case $(A_2A_3)_f = (A_2A_3)_i$ minus number of symbols counted.

$(B)_f = (B)_i$

If the operation is concluded in the Simultaneous Mode:

$(S_1)_f = (A_1)_i$

$(S_2S_3)_f = (0000)_8$ unless the BTC was reached, in which case $(S_2S_3)_f = (A_2A_3)_i$ minus the number of symbols counted.

Timing

Total time in milliseconds $= 3.575 + .03n + 4m$

where n is the total number of characters read, including symbols being counted, and m is the number of gaps encountered.

17: Rewind to BTC (RWD)

General Description

This instruction causes a designated magnetic tape to be completely rewound. Once the operation has been initiated, the rewind proceeds totally independent of the Computer, occupying neither the Normal nor the Simultaneous Mode. The Computer, after initiating the rewind, is free to execute other instructions.

Format

A address — ignored.

B address:

B_1 — specifies the address of the Tape Station.

B_2B_3 — ignored.

Direction of Tape Movement

Reverse.

Outline of Logic

A start signal is sent from the Computer to the specified Tape Station, whereupon the Computer is completely divorced from the operation; rewinding proceeds independently. If, while a tape is rewinding, it is selected by an instruction entailing forward movement, the instruction will not be executed until the current rewind operation has been completed (i.e., until the BTC has been reached). If an instruction involving backward movement of a tape is given while it is rewinding, or follows a completed RWD without an intervening operation involving forward movement of that tape, the instruction will merely go through STA (after the BTC is reached) and the Computer will proceed to the next instruction. The RWD instruction is not stored in standard HSM locations.

Addressable Registers Used

B

Final Register Contents

Immediately RWD is initiated, registers are available for use by the next instruction.

Timing

Total Computer time:

300 μs if the BTC is not positioned at the read-write head when the RWD instruction is given.

105 μs if the BTC is already positioned at the read-write head when the RWD instruction is given.

34: Sector Clear by Character (SCC)

General Description

This instruction places space characters in all the locations between, and including, two HSM addresses.

Format

A address — leftmost HSM location in the sector to be cleared.

B address — rightmost HSM location in the sector to be cleared.

Direction of Operation

Right to left.

Outline of Logic

The tetrad containing the rightmost character to be cleared, specified by the B Register, is read out of the HSM. This character, however, is inhibited from reaching the Memory Register. Instead, a space is generated and inserted into the empty location in the Memory Register. When regeneration occurs, the tetrad, with the inserted space, is returned to the HSM. The contents of the B Register are decreased by $(01)_8$ with each location cleared. This process is repeated until the leftmost character, specified by the A Register, is replaced with a space, so that the entire sector is cleared.

When HSM locations not included in the system are addressed, the Computer will perform as much of the instruction as possible and then continue on to the next instruction in sequence. Sector Clear by Character does not stop the Computer, with parity error notification, when it contains a programming error of this type.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (B)_i - n = (A)_i - (01)_8$

where n is the number of characters cleared.

Timing

Total time in microseconds = $15n$, where n is the total number of locations cleared.

Example (SCC)

Instruction: 34 036143 00 036152

HSM before Instruction is executed:

0361	41	42	43	44	45	46	47	50	51	52	53	54
	1	1	8	<	●	A	2	2	9	●	C	L

$(A)_i$ $(B)_i$

0361	41	42	43	44	45	46	47	50	51	52	53	54
	1	1	–	–	–	–	–	–	–	–	C	L

$(B)_f$ $(A)_f$

Final register contents:

$(A)_f = 036143$ $(B)_f = 036142$

Time:

15 x 8 = 120 μs.

36: Sector Clear by Tetrad (SCT)

General Description

This instruction inserts spaces $(01)_8$ in the HSM locations between, and including, two given tetrad addresses. It differs from the Clear Sector by Character Instruction (34) in that it clears four characters at a time, and is therefore four times faster.

Format

A address — leftmost tetrad to be cleared.

B address — rightmost tetrad to be cleared.

Direction of Operation

Right to left.

Outline of Logic

The characters in the rightmost tetrad to be cleared, specified by the B Register, are read out of the HSM, and four spaces are generated and inserted in their place.

The contents of the B Register are decreased by $(04)_8$ and the tetrad immediately to the left of the cleared tetrad is read out of the HSM. The process is repeated for each tetrad until the leftmost tetrad in the sector, specified by the A Register, is filled with spaces. The Computer will attempt to carry out the instruction even when HSM locations not included in the system are addressed, performing as much of the instruction as possible before continuing on to the next instruction in sequence. Clear Sector by Tetrad does not stop the Computer, with parity error notification, when it contains a programming error of this type.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$ $(B)_f = (B)_i - 4n$

where n is the number of tetrads cleared.

Timing

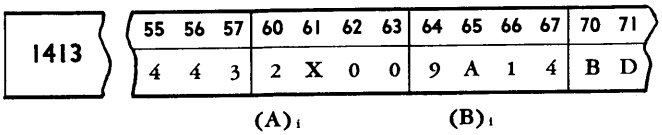
Total time in microseconds = $15n$

where n is the number of tetrads cleared.

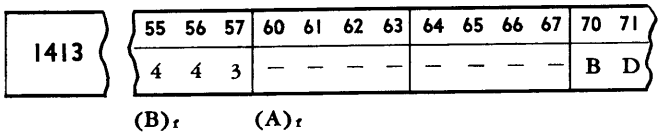
Example (SCT)

Instruction: 36 141360 00 141365

HSM before Instruction is executed:



HSM after Instruction:



Final register contents:

$(A)_f = 141360$ $(B)_f = 141355$

Time:

15 x 2 = 30 μs.

22: One-Character Transfer (OCT)

General Description

This instruction transfers the contents of one HSM location into another HSM location. It may be used to modify portions of instructions, transfer one-character constants, etc.

Format

A address — HSM location of character to be transferred.

B address — HSM destination location.

Outline of Logic

The character stored in the location specified by the A Register is transferred to the location specified by the B Register, which terminates the operation.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (B)_i$

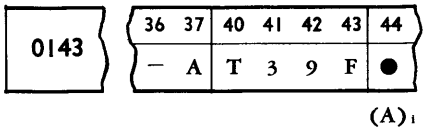
Timing

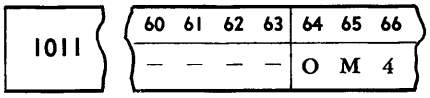
30 μs.

Example (OCT)

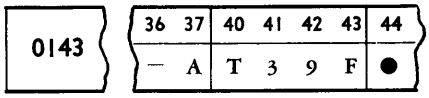
Instruction: 22 014344 00 101163

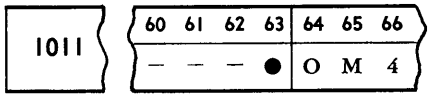
HSM before Instruction is executed:





HSM after Instruction:





Final register contents:

$(A)_f = 014344$

$(B)_f = 101163$

Time:

30μs.

25: Three-Character Transfer (TCT)

General Description

This instruction transfers, in parallel, the contents of the rightmost three locations of one tetrad to the rightmost three locations of another tetrad in the HSM. It is useful for placing addresses into stored instructions, setting Address Modifiers, etc.

Format

The A and B addresses need not refer to the same relative positions, and they may refer to any of the locations in the tetrads.

A address — HSM address of the tetrad which contains, in C_1, C_2 and C_3, the characters to be transferred.

B address — HSM address of the destination tetrad.

Direction of Operation

Parallel transfer of all three characters.

Outline of Logic

The contents of the tetrad addressed by the A Register are transferred to the Memory Register, from which the rightmost three characters (C_1, C_2, and C_3) are admitted to, and retained in, the Bus Adder. The tetrad

addressed by the B Register is then read out of the HSM, but only the leftmost character (C_0) enters the Memory Register. The contents of the Bus Adder are returned to the Memory Register, the entire contents of which are transferred to the tetrad addressed by the B Register, completing the operation. The destination tetrad now contains the C_1, C_2, and C_3 characters of the original tetrad; its C_0 character remains unaltered.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$

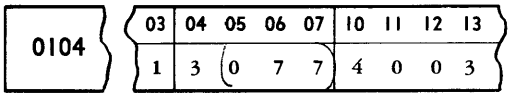
Timing

30 μs.

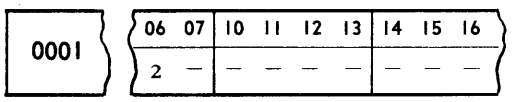
Example (TCT)

Instruction. 25 010406 00 000111

HSM before Instruction is executed:

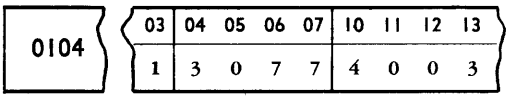


$(A)_i$

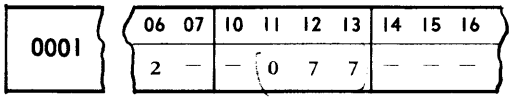


$(B)_i$

HSM after Instruction:



$(A)_f$



$(B)_f$

Final register contents:

$(A)_f = 010406$ $(B)_f = 000111$

Time:

30 μs.

24: Sector Transfer by Character (STC)

General Description

This instruction transfers a series of characters from an area (sector) between, and including, two designated HSM locations to another HSM area (sector).

Format

A address — HSM location of leftmost character of the series to be transferred.

B address — HSM location of rightmost character of the series to be transferred.

Preset T Register [see Set Register instruction (72), page 23], so that it holds the rightmost destination location.

Direction of Operation

Right to left.

Outline of Logic

The character stored in the location specified by the B Register is placed in the location specified by the T Register. $(01)_8$ is subtracted from the B Register and from the T Register. These steps are repeated until the character in the location specified by the A Register has been processed.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$ $(B)_f = (A)_i - (01)_8$
$(T)_f = (T)_i - n$
where n = the number of characters transferred.

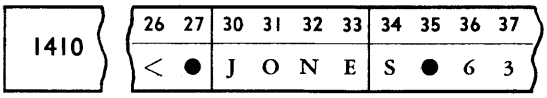
Timing

Total time for the operation in microseconds = $30n$ where n is the number of characters transferred.

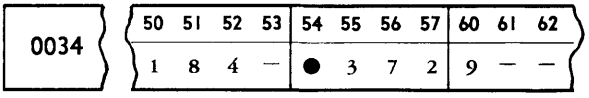
Example (STC)

Instruction: 24 141027 00 141034
T Register contains 003461

HSM before Instruction:

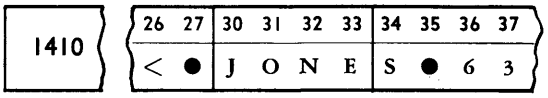


$(A)_i$ $(B)_i$

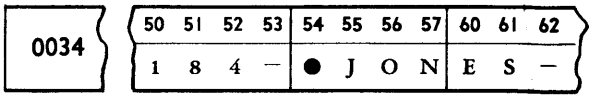


$(T)_i$

HSM after Instruction:



$(B)_f$ $(A)_f$



$(T)_f$

Final register contents:

$(A)_f = 141027$ $(B)_f = 141026$ $(T)_f = 003453$

Time:

30 x 6 = 180 μs.

26: Sector Transfer by Tetrad (STT)

General Description

This instruction transfers the contents of one tetrad or any number of consecutive HSM tetrads between, and including, two specified tetrad addresses, into another specified tetrad or consecutive series of tetrads. This instruction differs from Sector Transfer by Character (24) in that it transfers four characters at a time, and is therefore four times faster.

Format

A address — specifies the address of the leftmost tetrad of the sector to be transferred.

B address — specifies the address of the rightmost tetrad of the sector to be transferred.

Preset T Register [see Set Register instruction (72), page 23], so that it holds the address of the rightmost destination tetrad.

Direction of Operation

Right to left.

Outline of Logic

The rightmost tetrad to be transferred, addressed by the B Register, is read out of the HSM and into the Memory Register and is retained there. The tetrad specified by the T Register is then read out of the HSM but is inhibited from reaching the Memory Register. The contents of the Memory Register are placed in the tetrad designated by the T Register, thus transferring the rightmost tetrad of the sector into its destination location. The contents of the B and T Registers are each decreased by $(04)_8$ and the process is repeated until the tetrad addressed by the A Register (leftmost tetrad in the sector) has been processed.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$ $(B)_f = (B)_i - 4n$

$(T)_f = (T)_i - 4n$

where n is the number of tetrads transferred.

Timing

Total time in microseconds = $30n$

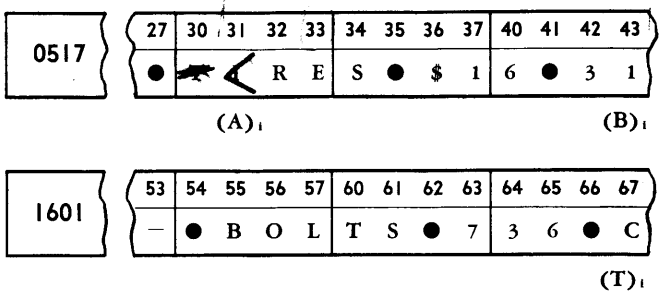
where n is the number of tetrads transferred.

Example (STT)

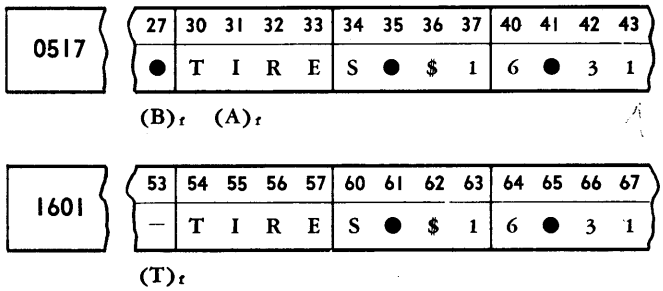
Instruction: 26 051731 00 051743

T Register contains 160167

HSM before Instruction is executed:



HSM after Instruction:



Final register contents:

$(A)_f = 051731$ $(B)_f = 051727$ $(T)_f = 160153$

Time:

30 x 3 = 90 μs.

21: Item Transfer (IT)

General Description

This instruction transfers an item from one group of successive HSM locations to another.

Format

A address — HSM location of the rightmost character of the item to be transferred.

B address — rightmost destination location.

Direction of Operation

Right to left.

Outline of Logic

The character stored in the location specified by the A Register is read out of the HSM. It is then placed in the HSM location specified by the B Register. The contents of the A and B Registers are each decreased by $(01)_8$, so that they now address the HSM locations immediately to the left of the processed locations. The operation ends when an ISS has been transferred.

If the item to be transferred contains only spaces (to the right of the ISS), PRN is set. If it contains one or more non-space characters, PRP is set.

If the designated destination area overlaps the location of the ISS in the original area, all the contents of the HSM to the left of the location where overlap began will be destroyed.

Addressable Registers Used

A B

Final Register Contents

$(A)_f$ = (HSM location of ISS in the original area) $- (01)_8$

$(B)_f$ = (HSM location of ISS in the destination area) $- (01)_8$

Timing

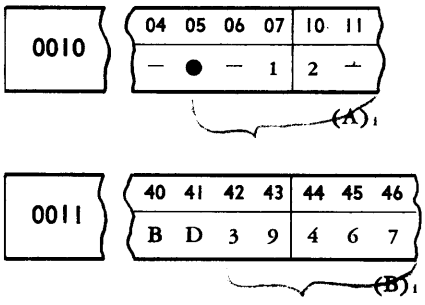
Total time in microseconds = $30n$

where n is the number of characters transferred.

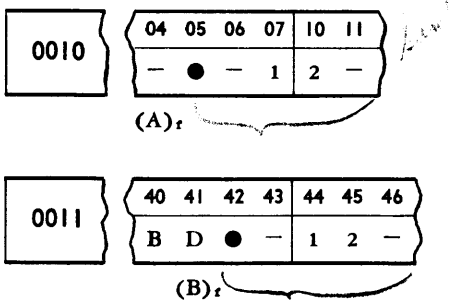
Example (IT)

Instruction: 21 001011 00 001146

HSM before Instruction is executed:



HSM after Instruction:



Final register contents:

$(A)_f$ = 001004 PRP is set.

$(B)_f$ = 001141

Time:

30 x 5 = 150 μs.

31: Locate n^{th} Symbol in Sector (LNS)

General Description

This instruction searches through the contents of successive HSM locations between, and including, two given addresses, counting the occurrences of a designated symbol. The operation ceases when (1) the specified count is reached or (2) the rightmost location in the sector has been searched.

Format

A address — leftmost HSM location in sector to be searched.

B address — rightmost HSM location in sector to be searched.

Preset T Register [see Set Register instruction (72), page 23] so that

T_1 designates the symbol. [Symbol cannot be $(00)_8$.]

T_2T_3 specifies the desired count (octal).

Direction of Operation

Left to right.

Outline of Logic

The T_1 character is sent to the L Register. The character in the HSM location addressed by the A Register is sent to the R Register and the contents of the A Register are increased by $(01)_8$. Each time the L and R registers compare equal, T_2T_3 is decreased by $(01)_8$. The operation ends (1) when the end of the sector is reached (i.e., when the character in the rightmost location has been examined, and the contents of the A Register are equal to those of the B Register) or (2) when T_2T_3 is decreased to $(0000)_8$. The A Register is then decreased by $(01)_8$, so that it finally contains the address of the HSM location immediately to the left of (1) the rightmost location in the sector searched or (2) the n^{th} symbol.

The PRI's are terminally set as follows:

PRZ if $(T_2T_3)_f = (0000)_8$ and $(A)_f \neq (B)_f$

PRP if $(T_2T_3)_f = (0000)_8$ and $(A)_f = (B)_f$

PRN if $(T_2T_3)_f \neq (0000)_8$ and $(A)_f = (B)_f$

If $(T_2T_3)_i$ is $(0000)_8$, the instruction is not performed, the PRI's are set (see below) and the next instruction in sequence is performed.

If $(T_2T_3)_i$ is $(0000)_8$ and $(A)_i = (B)_i$ or $(A)_i \neq (B)_i$, PRZ is set.

If $(T_2T_3)_i > (0000)_8$ and $(A)_i = (B)_i$, PRN is set.

Addressable Registers Used

A B T (preset)

Final Register Contents

(1) when the operation is terminated by $(T_2T_3)_f = (0000)_8$

$(A)_f = (A)_i + n - (02)_8$, where n is the number of locations searched, or, restated,

$(A)_f$ = HSM location of n^{th} symbol minus $(01)_8$.

$(B)_f = (B)_i$

$(T_1)_f = (T_1)_i$

(2) when the operation is terminated by A = B

$(A)_f = (B)_i - (01)_8$

$(B)_f = (B)_i$

$(T_1)_f = (T_1)_i$

$(T_2T_3)_f = (T_2T_3)_i$ minus number of occurrences counted.

Timing

If $(T_2T_3)_i > (0000)_8$ and $(A)_i < (B)_i$:

$15m + 30n + 45$ μs

where m = total number of locations searched, and n = total number of occurrences counted.

If $(T_2T_3)_i = (0000)_8$, time is 15 μs.

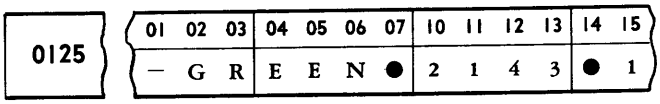
If $(A)_i = (B)_i$, time is 60 μs.

Example (LNS)

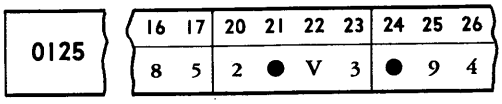
Instruction: 31 012506 00 012524

T (preset) = 740003

HSM before and after execution of Instruction:



$(A)_i$



$(A)_f$ $(B)_i$ $(B)_f$

Final register contents:

$(A)_f = 012520$ $(B)_f = 012524$ $(T)_f = 740000$

PRZ is set.

Time:

(15 μs x 12) + (30 μs x 3) + 45 μs = 315 μs.

27: Random Distribute (RD)

General Description

This instruction redistributes successive items in the HSM, to locations designated by a stored list of addresses.

Format

A address — HSM location of leftmost character of leftmost item to be dispersed. (The first character to be dispersed will be an ISS or an SM.)

B address — refers to the stored list of addresses; it specifies the address of the tetrad which contains the destination address for the SM or the ISS of the leftmost item to be dispersed.

The list is comprised of successive tetrads which contain the destination addresses in their rightmost three locations (C_1, C_2, and C_3); the C_0 character in each of these tetrads is ignored. When the destination address is $(777777)_8$, the associated item is not transferred. The destination address $(000000)_8$ terminates the list.

Direction of Operation

Left to right.

Outline of Logic

The leftmost character of the leftmost item, addressed by the A Register, is read out of the HSM and into the R Register. Unless this character is an ISS, SM, or EM, it will not be further processed. [The contents of the A Register are increased by $(01)_8$, whether or not the character is transferred.] Instead, the next character to the right is tested. If this is an ISS or SM, the tetrad designated by the B Register is read (from the list) into the Memory Register. The rightmost three characters of this tetrad are placed in the T Register. If these characters are not $(777777)_8$ or $(000000)_8$, successive transfer of characters begins, starting with the ISS or SM, and continues until another ISS or SM is found. The contents of the B Register are increased by $(04)_8$ and the address contained in the next (to the right) tetrad in the list is then placed in the T Register, and the process is repeated.

When an EM is found, an ISS (rather than EM) is placed in the location designated by the T Register. If, when the EM is found, the list has not been exhausted, ISS's will also be placed in the remaining destination locations designated in the list.

When the destination address contained in the list is $(777777)_8$ the associated item is *not* transferred; instead, characters are read out successively [and the contents of the A Register are increased by $(01)_8$ for each character] until another ISS or SM is found.

The operation is completed when the terminal address, $(000000)_8$, is found in the list.

If the terminal address $(000000)_8$ is sensed prior to an EM, PRZ is set. If both the terminal address and an

EM have been sensed, PRP is set. If an EM has been sensed prior to the terminal address, PRN is set.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f$ = HSM address of last control symbol (ISS, SM, or EM) sensed in the original area.

$(B)_f = (B)_i + 4n$

where n is the number of addresses in the list (including terminal and "throw away" addresses).

$(T)_f = (000000)_8$

Timing

Total time in microseconds $\doteq 33n_1 + 18n_2 + 45n_3$

where: n_1 is the total number of characters transferred; n_2 is the total number of characters whose distribution address is $(777777)_8$; n_3 is the number of distribution addresses left when an EM is found. (The address of the EM must be included in n_3.)

Example (RD)

Instruction: 27 002604 00 140134

HSM before and after Instruction is executed:

	02	03	04	05	06	07	10	11	12	13	14	15	16
0026	B	Z	<	●	D	O	E	–	J	–	J	●	A

$(A)_i$

	17	20	21	22	23	24	25	26	27	30	31	32	33
0026	3	3	0	6	2	●	Q	●	1	2	–	0	2

	34	35	36	37	40	41	42	43	44	45	46	47
0026	–	3	5	●	–	1	5	0	0	>	A	>

$(A)_f$

	34	35	36	37	40	41	42	43	44	45	46	47	50
1401	60	77	77	77	10	77	77	77	01	00	10	36	01

$(B)_i$

	51	52	53	54	55	56	57	60	61	62	63	64	65
1401	00	10	50	74	00	10	55	77	77	77	77	77	00

	66	67	70	71	72	73	74	75	76	77
1401	10	66	51	00	10	74	01	00	00	00

HSM locations 001036 – 001077 have been cleared to spaces before Instruction is executed.

HSM after Instruction:

	36	37	40	41	42	43	44	45	46	47	50	51	52
0010	●	A	3	3	0	6	2	–	–	–	●	Q	–

	53	54	55	56	57	60	61	62	63	64	65	66	67
0010	–	–	●	1	2	–	0	2	–	3	5	●	–

	70	71	72	73	74	75	76	77
0010	–	–	–	–	●	–	–	–

Final register contents:

$(A)_f$ = 002645 $(B)_f$ = 140200 $(T)_f$ = 000000

PRN is set

Time:

(33 x 20) + (18 x 15) + (45 x 2) = 1020 μs.

32: Zero Suppress (ZS)

General Description

This instruction is used to delete the non-significant zeros to the left of the MSC of the result of a decimal arithmetic operation.

Format

A address — leftmost HSM location in sector in which the zeros are to be suppressed.

B address — rightmost HSM location in sector in which the zeros are to be suppressed.

Direction of Operation

Left to right.

Outline of Logic

The character in the location specified by the A Register is read out of the HSM, and the contents of the A Register are transferred to the T Register. If this character is a space, it remains unchanged; if it is a zero, it is replaced by a space. The contents of the A Register are increased by $(01)_8$ with each location searched. The contents of the T Register are increased by $(01)_8$ with each location processed. These steps are repeated until (1) a space is found after at least one zero has been suppressed, or (2) a character which is neither a space nor a zero is found, or (3) the contents of the A and B Registers are equal. At the conclusion of ZS, the A Register is reset as shown under Final Register Contents.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f = (A)_i + n - (02)_8$

$(B)_f = (B)_i$

$(T)_f = (A)_f + (01)_8$

where n is the number of locations searched.

If ZS is terminated by A-B equality:

a. *Only zeros encountered*

$(A)_f = (B)_f = (B)_i$

$(T)_f = (B)_i + (01)_8$

b. *Only spaces encountered*

$(A)_f = (B)_f = (T)_f = (B)_i$

Timing

$$15m + 30n + 15\ \mu s$$

If ZS is terminated by A-B equality:

$$15m + 30n$$

where m is the number of spaces preceding the first non-space character and n is the number of zeros suppressed.

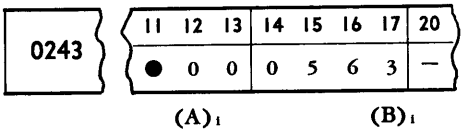
If no zeros or spaces were found:

$$30\ \mu s$$

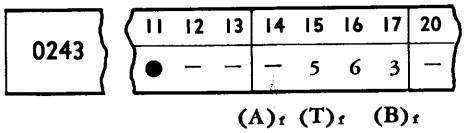
Examples (ZS)

1. *Instruction:* 32 024312 00 024317

HSM before Instruction is executed:



HSM after Instruction:

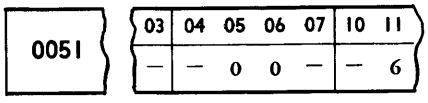


Time:

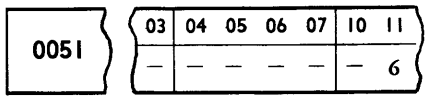
(30 μs x 3) + 15 μs = 105 μs.

2. *Instruction:* 32 005103 00 005111

HSM before Instruction is executed:



HSM after Instruction:

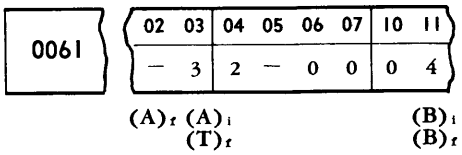


Time:

(15 μs x 2) + (30 μs x 2) + 15 = 105 μs.

3. *Instruction:* 32 006103 00 006111

HSM before and after Instruction is executed:

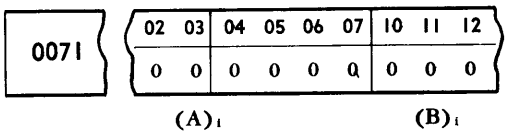


Time:

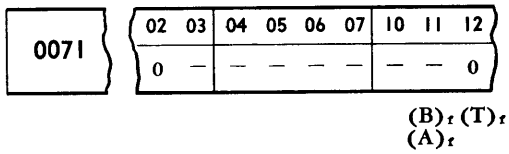
30 μs.

4. *Instruction:* 32 007103 00 007111

HSM before Instruction is executed:



HSM after Instruction:

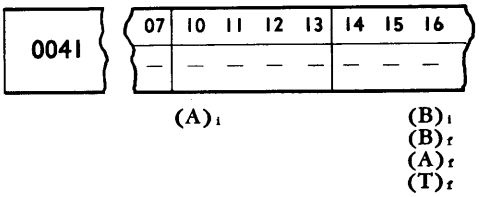


Time:

(30 μs x 7) = 210 μs.

5. *Instruction:* 32 004110 00 004116

HSM before and after Instruction is executed:



Time:

(15 μs x 7) = 105 μs.

33: Justify Right (JR)

General Description

This instruction, used to effect right columnar alignment, (1) adjusts and transfers an item from one series of successive HSM locations to another, or (2) adjusts the item, leaving it in the same group of HSM locations. All the space symbols which were originally located to the right of the sign position, are placed between

the ISS and the MSD in the destination area. (The sign position is the HSM location immediately to the right of the LSD.)

Format

A address — rightmost HSM location of the item to be justified.

B address — destination location of the sign of the item.

If the A and B addresses are the same and refer to a non-space, non-minus character, a sign will *not* be generated. (In this case, no change in the item is effected by the JR instruction.)

If the A and B addresses are not the same, and the rightmost character of the item is neither a space nor a minus, a space (which in the sign position is interpreted as a plus sign) will be generated in the destination (B address) location.

If the B address falls between the A address and the original location of the ISS, all of memory to the left of, and including, the B address will be destroyed.

Direction of Operation

Right to left.

Outline of Logic

Beginning with the HSM location specified by the A Register, and proceeding to the left, the characters in the item are successively processed. With each space or minus sign found, $(01)_8$ is added to the contents of the T Register [which was automatically set to $(000000)_8$ during staticizing]. When the first non-space, non-minus character is found, the sign is generated in the location specified by the B Register and the contents of the T Register are decreased by $(01)_8$. The sign of the item will be minus if a minus sign is sensed in any of the locations between the A address and the rightmost non-minus, non-space character. The remaining characters addressed by the A Register are then transferred into the locations specified by the B Register. The A and B Registers are decreased by $(01)_8$ with each location processed. When the A Register addresses a location which contains an ISS, space symbols equal in number to the contents of the T Register are generated in the destination locations immediately to the left of the MSD. The ISS is then transferred, completing the operation.

If the item contains only spaces and an ISS or only an ISS, PRN is set. If it contains any non-space character (in addition to the ISS), PRP is set.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f$ = (original location of ISS) − $(01)_8$

$(B)_f$ = (destination location of ISS) − $(01)_8$

$(T)_f$ = $(000000)_8$ if first location examined contains a space or a minus sign.

= $(777777)_8$ if first location examined contains a non-space, non-minus character.

Timing

Total time in microseconds = $30\ (n + m)$

where n = number of space and/or minus characters to the right of the first non-minus, non-space character encountered.

where m = number of non-space, non-minus characters transferred. (m includes the ISS.)

Example (JR)

Instruction: 33 001433 00 154172

HSM before Instruction:

	22	23	24	25	26	27	30	31	32	33	34	35
0014	X	●	1	3	6	9	4	⊖	−	−	−	Y

$(A)_i$

	61	62	63	64	65	66	67	70	71	72	73	74	75
1541	−	W	A	S	H	I	N	G	T	O	N	−	D

$(B)_i$

HSM after Instruction:

	22	23	24	25	26	27	30	31	32	33	34	35
0014	X	●	1	3	6	9	4	⊖	−	−	−	Y

$(A)_f$

	61	62	63	64	65	66	67	70	71	72	73	74	75
1541	−	●	−	−	1	3	6	9	4	⊖	N	−	D

$(B)_f$

Final register contents:

$(A)_f = 001422$ $(B)_f = 154161$ $(T)_f = 000000$

PRP is set.

Time:

$(30 \times 3) + (30 \times 6) = 270\ \mu s.$

35: Sector Compress — Retain Redundant ISS's (SCR)

General Description

This instruction transfers a sector of characters from one part of the HSM to another, removing, in the process, all spaces located to the right of the rightmost non-space character within each item in the sector. ✱Note: A space in the sign position (i.e., positive sign) is also deleted.

Format

A address — leftmost HSM location of the sector to be compressed and transferred.

B address — rightmost HSM location of the sector to be compressed and transferred.

Preset T Register [see Set Register instruction (72), page 23], so that it holds the rightmost destination address.

Direction of Operation

Right to left.

Outline of Logic

The rightmost character of the sector, specified by the contents of the B Register, is read out of the HSM and stored in the R Register. The contents of the B Register are decreased by $(01)_8$. If the character is a space symbol, it is not transferred; instead, the next character to the left is read out and tested. When a non-space character is found, it is transferred to the location specified by the T Register. The Computer then continues sequential transfer of all the characters to the left of this location until an ISS or EM is transferred. [The contents of the T Register are decreased by $(01)_8$ with each transfer.] Beginning with the location to the left of the ISS or EM, the Computer again tests for space symbols and does not resume transferring until the next non-space character is found. These steps are repeated until the leftmost character of the sector, specified by the A Register, has been processed.

If the sector contains *any* non-space, non-EM, non-ISS characters, PRP is set. If it contains *only* space, EM, and/or ISS characters, PRN is set.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$ $(B)_f = (A)_i - (01)_8$

$(T)_f = (T)_i - n$

where n is the number of characters actually transferred.

Timing

Total time in microseconds $= 15n + 15m$

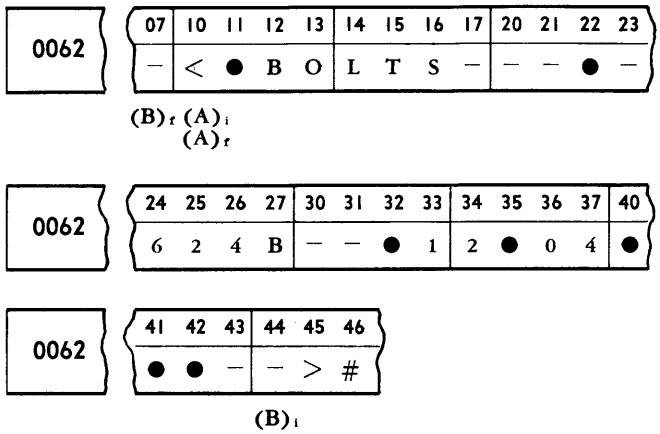
where n is the number of characters actually transferred, and m is the total number of characters in the original sector.

Example (SCR)

Instruction: 35 006210 00 006244

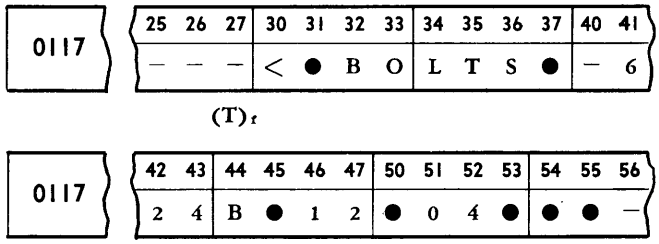
T Register contains 011755

HSM before and after Instruction:



Assume the destination area contains only spaces before the instruction is executed.

HSM after Instruction:



Final register contents:

$(A)_f = 006210$ $(B)_f = 006207$ $(T)_f = 011727$

Time:

(15 μs x 22) + (15 μs x 29) = 765 μs.

37: Sector Compress — Delete Redundant ISS's (SCD)

General Description

This instruction transfers a sector of data from one part of the HSM to another, deleting, in the process, (1) all ISS's originally located to the right of the rightmost non-ISS, non-space character in the sector, and (2) all spaces located to the right of the rightmost non-space character within each item in the sector. ✱Note: A space in the sign position (i.e., positive sign) is also deleted.

Format

A address — leftmost HSM location of sector to be compressed and transferred.

B address — rightmost HSM location of sector to be compressed and transferred.

Preset T Register [see Set Register instruction (72), page 23], so that it holds the rightmost destination address.

Direction of Operation

Right to left.

Outline of Logic

The rightmost character of the sector to be compressed is addressed with the contents of the B Register, and then tested. If it is an ISS or a space symbol, it is not transferred; instead the next character to the left is read out and tested. When a non-space, non-ISS character is found, it is transferred to the location specified by the T Register. From this point on, until the leftmost character of the sector, specified by the A Register, has been processed, transfer is continuous, excluding groups of successive spaces immediately to the left of ISS's but including all the remaining ISS's in the sector.

Note: If the B address is that of the EM or of a location to the right of the EM, redundant ISS's to the left of the EM will not be deleted.

If the sector contains *any* non-space, non-ISS characters, PRP is set. If it contains *only* spaces and/or ISS characters, PRN is set.

Addressable Registers Used

A B T (preset)

Final Contents

$(A)_f = (A)_i$ $(B)_f = (A)_i - (01)_8$

$(T)_f = (T)_i - n$

where n is the number of characters actually transferred.

Timing

Total time in microseconds $= 15n + 15m$

where n is the number of characters actually transferred and m is the total number of characters in the original sector.

Example (SCD)

Instruction: 37 076105 00 076133

T Register contains 076133

HSM before Instruction is executed:

0761	05	06	07	10	11	12	13	14	15	16	17	20	21
	●	S	M	I	T	H	–	–	●	1	9	–	P

$(A)_i$

0761	22	23	24	25	26	27	30	31	32	33	34
	I	N	E	–	–	–	●	●	●	●	>

$(B)_i$
$(T)_i$

HSM after Instruction:

0761	04	05	06	07	10	11	12	13	14	15	16	17	20
	–	●	S	M	I	T	H	–	–	●	●	S	M

$(B)_f$ $(A)_f$ $(T)_f$

0761	21	22	23	24	25	26	27	30	31	32	33	34
	I	T	H	●	1	9	–	P	I	N	E	>

Final register contents:

$(A)_f = 076105$ $(B)_f = 076104$ $(T)_f = 076115$

Time:

(15 μs x 14) + (15 μs x 23) = 555 μs.

51: Decimal Add (DA)

General Description

This instruction performs decimal addition, in accordance with algebraic rules, producing a left-justified, non-zero-suppressed sum, which is stored in the HSM locations originally occupied by the augend. The operands may be of any length and, also, of unequal lengths.

Format

A address — HSM location of the rightmost character (sign or space to the right of the sign) of the augend (and sign of the sum).

B address — HSM location of the rightmost character (LSD, or sign, or space to the right of the sign) of the addend.

(The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS.)

Direction of Operation

Right to left.

Outline of Logic

The initial contents of the A Register (HSM address of the rightmost character of the augend) are transferred to the T Register, which is used thereafter to place the sum in the HSM. Next, a search is made in each operand for the rightmost non-space, non-minus character. The contents of the A Register are decreased by $(01)_8$ with each location searched in the augend; the contents of the B Register are decreased by $(01)_8$ with each location searched in the addend.

The search is concluded for each operand when the character addressed is neither space nor minus.

The sign of the sum is then placed in the HSM location specified by the contents of the T Register. The contents of the T Register are then decreased by $(01)_8$. If the signs are unlike, the sum is initially assumed to be positive and is changed later, if necessary. The searching process, with decrease of A and B Register contents, permits right column alignment of the operands so that the first addition is performed on the least significant digits.

After a sign has been stored, addition takes place as follows:

1. If the signs of the operands are unlike, the digits of the negative operand are complemented (tens complement) before addition takes place.
2. The tetrad containing the LSD of the augend is read out of the HSM and into the Memory Register, and the contents of the A Register are decreased by $(01)_8$. If this character is not a control symbol, it is transferred to, and retained in, the L Register in the Arithmetic Unit.
3. The tetrad containing the LSD of the addend is read out of the HSM and into the Memory Register, and the contents of the B Register are decreased by $(01)_8$. If this character is not a control symbol, it is transferred to the R Register in the Arithmetic Unit.
4. The contents of the L Register are added to those of the R Register, and the LSD of the sum is placed in the HSM location specified by the T Register. The contents of the T Register are then decreased by $(01)_8$. The carry, if any, is retained in the Arithmetic Unit.

The process described above is repeated until the end of either operand is reached. An operand is considered ended when an ISS is sensed or when a space to the left of the MSD is sensed.

If the operands were of unequal length, the carry (if any) is added to the next digit(s) of the longer operand. When there is no carry, a zero is added to each of the remaining digits of the longer operand to complete the addition.

If the signs of the operands were alike, or if they were unlike and the sum is positive, the operation is concluded with an ISS placed in the HSM location immediately to the left of the MSD of the sum. However, if the signs of the operands are unlike and the sum is negative, the sum is re-complemented and the initially assumed positive sign is changed to negative. An ISS is placed to the left of the sum as in the above cases.

If the A Register initially addresses an ISS and the B Register initially addresses the LSD or the sign of the addend, the addend will be transferred into the augend (sum) locations, with the sign stored in the location that originally held the ISS of the augend. In this case, if the LSD of the addend is initially addressed, the sign will be assumed to be plus. If both the A and the B Registers initially address an ISS, a space (plus sign) will be stored in the location originally occupied by the ISS of the augend, followed by an ISS immediately to the left.

If both operands contain only space and/or minus characters to the right of the ISS, the sign will be placed in the location initially addressed by the A (and T) Register, and an ISS in the location immediately to the left of the sign.

Non-significant zeros are not suppressed by this instruction. When the result of an addition is zero, zeros will appear in the sum locations, and the sign will be positive. The PRI's are set after the instruction according to the sign of the sum, except that a zero result will set PRZ.

Note: The first character addressed in the augend must be a space, minus or ISS. If it is any other character, an alarm stop occurs.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f$ = the HSM location immediately to the left of the ISS or the space that terminates the augend.

$(B)_f$ = the HSM location immediately to the left of the ISS or the space that terminates the addend.

$(T)_f$ = the HSM location of the ISS of the sum.

Timing

Time in microseconds = $15n_1 + 45n_2 + 30n_3 + 90$

n_1 is the total number of space and/or minus characters found to the right of both operands;

n_2 is the number of digits in the shorter operand;

n_3 is the difference in number of digits of the operands.

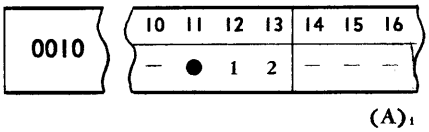
For negative sum, add $30\ (n + 1) + 15$

where n = number of digits in the sum.

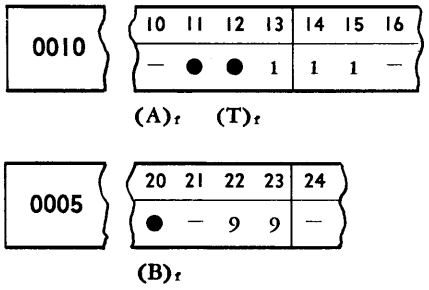
Examples (DA)

1. *Instruction:* 51 001016 00 000524

HSM before Instruction is executed:



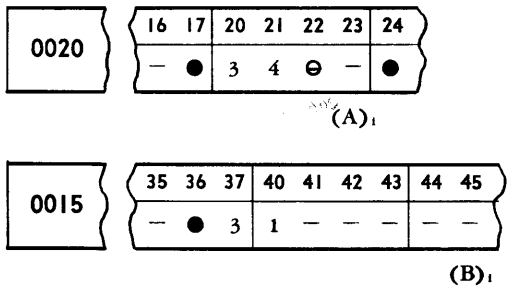
HSM after Instruction:



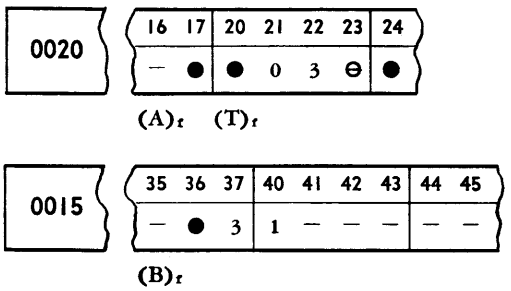
Time:
(15 x 4) + (45 x 2) + 90 = 240 μs.
PRP is set.

2. *Instruction:* 51 002023 00 001545

HSM before Instruction is executed:



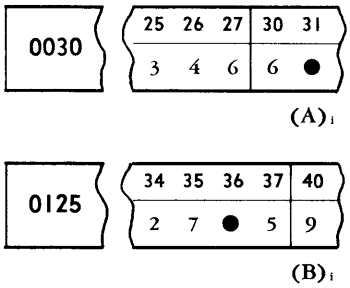
HSM after Instruction:



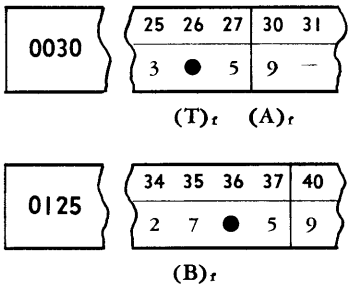
Time:
(15 x 7) + (45 x 2) + 90 + [30 (2 + 1)] + 15 = 390 μs.
PRN is set.

3. *Instruction:* 51 003031 00 012540

HSM before Instruction is executed:



HSM after Instruction:



Time:
(15 x 0) + (45 x 0) + (30 x 2) + 90 = 150 μs.
PRP is set.

52: Decimal Subtract (DS)

General Description

This instruction performs decimal subtraction on variable length operands. The subtraction is algebraic. Specifications as to operands, storage of the difference and end conditions are exactly the same as in Decimal Add.

Format

A address — HSM location of the rightmost character (sign or space to the right of sign) of the minuend (and sign of the difference).

B address — HSM location of the rightmost character (LSD, or sign, or space to the right of the sign) of the subtrahend.

(The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS.)

Direction of Operation

Right to left.

Outline of Logic

The operands are searched for the least significant non-space, non-minus character.

The sign of the subtrahend is reversed.

If the original signs of the operands are unlike, the sign of the minuend is the sign of the difference. If the signs are alike, the sign is assumed positive and changed later if necessary.

From this point Decimal Subtract is exactly the same as Decimal Add.

✱Note: The first character addressed by the A Register must be a space symbol, minus sign or ISS. If it is any other character, an alarm stop occurs.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f$ = the HSM location immediately to the left of the ISS or the space that terminates the minuend.

$(B)_f$ = the HSM location immediately to the left of the ISS or the space that terminates the subtrahend.

$(T)_f$ = the HSM location of the ISS of the difference.

Timing

Time in microseconds = $15n_1 + 45n_2 + 30n_3 + 90$

where n_1 = total number of spaces and/or minuses to the right of both operands.

n_2 = number of digits in shorter operand.

n_3 = difference in number of digits of the operands.

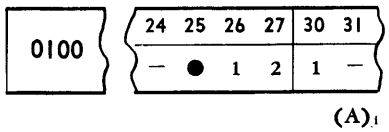
For negative results, add 30 $(n + 1) + 15$ μs.

where n = number of digits in the difference.

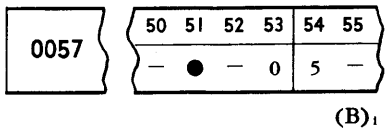
Examples (DS)

1. *Instruction:* 52 010031 00 005755

HSM before Instruction is executed:

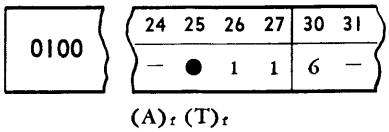


$(A)_i$

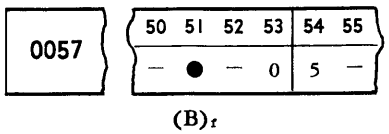


$(B)_i$

HSM after Instruction:



$(A)_f$ $(T)_f$



$(B)_f$

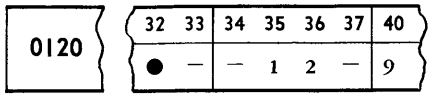
Time:

(15 x 2) + (45 x 2) + (30 x 1) + 90 = 240 μs.

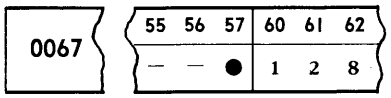
PRP is set.

2. *Instruction:* 52 012037 00 006761

HSM before Instruction is executed:

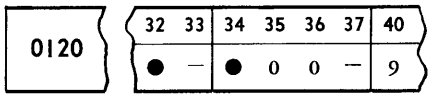


$(A)_i$

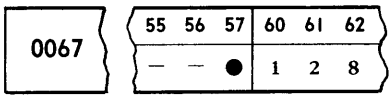


$(B)_i$

HSM after Instruction:



$(A)_f$ $(T)_f$



$(B)_f$

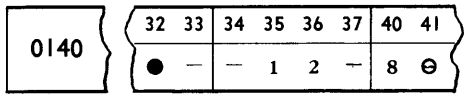
Time:

(15 x 1) + (45 x 2) + (30 x 0) + 90 = 195 μs.

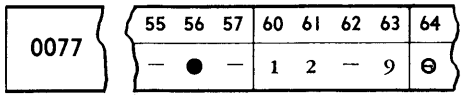
PRZ is set.

3. *Instruction:* 52 014041 00 007763

HSM before Instruction is executed:

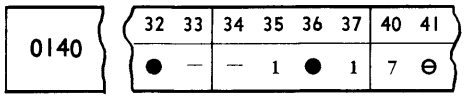


$(A)_i$

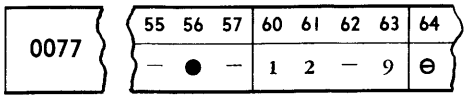


$(B)_i$

HSM after Instruction:



$(A)_f$ $(T)_f$



$(B)_f$

Time:

$(15 \times 1) + (45 \times 1) + 90 + [30\ (2 + 1)] + 15 = 255\ \mu s.$

PRN is set.

53: Decimal Multiply (DM)

General Description

This instruction performs decimal multiplication in accordance with algebraic rules, producing a right justified, non-zero-suppressed product. If a quantity is prestored in the product area, the product is added (absolute addition) to it, permitting round-off by any number and multiply-accumulate. However, the sign of the product will be assigned to the accumulated (absolute) result.

Format

A address — HSM location of the rightmost character (LSD, sign, or space to the right of the sign) of the multiplicand.

B address — HSM location of the rightmost character (LSD, sign, or space to the right of the sign) of the multiplier.

Preset T Register [see Set Register instruction (72), page 23], so that it contains the HSM location that is to receive the sign of the product.

✱ The product may not be stored in the HSM locations of the operands.

The length of each operand is defined by the first space to the left of a non-space, non-minus character or by an ISS. A space character or an ISS must be prestored in the product area in the location immediately to the left of the anticipated MSD.

Direction of Operation

Right to left.

Outline of Logic

The operands are searched for their LSD's, the contents of the A and B Registers being respectively decreased by $(01)_8$ with each location searched. A sign is then placed in the product location and the contents of the T Register are decreased by $(01)_8$. The sign of the product will be plus if the operand signs are alike, and minus if the signs are unlike. If an operand is not accompanied with a sign, it is assumed to be plus.

Multiplication is performed by a series of successive additions and shifts. The multiplicand is added to the contents of the product locations a number of times which is equal to the values of the digits of the multiplier.

The only characters not treated as numeric are minus sign, space symbol and ISS.

If the multiplier is all zeros, (and there is no prestored quantity in the product area), the product will be exactly like the multiplier (e.g., 634762 x 0000 = 0000).

If the multiplicand is all zeros, the number of zeros in the product will be equal to the number in the multiplicand plus the number of digits in the multiplier minus one.

If in either or both operands, an ISS is sensed before any other non-space, non-minus character, a plus sign is placed in the sign location of the product area, PRZ is set, and the operation ends.

If round-off or accumulate is desired, the prestored quantity must be properly positioned in the product area, since there is no search of the product before addition begins. If round-off or accumulate is not desired, the product area must be cleared to spaces.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f$ = HSM location of MSD of multiplicand minus $(02)_8$.

$(B)_f$ = HSM location of MSD of multiplier minus $(02)_8$.

$(T)_f$ = HSM location of MSD of product minus $(01)_8$.

Timing

Total time in milliseconds:

a. If $n_1 > 0$ and $n_2 > 0$

$$.015\ [10 + (12n_1 + 32)n_2] + .015n_3$$

b. If $n_1 = 0$ and $n_2 > 0$

$$.015\ (n_2 + n_3 + 3)$$

c. If $n_2 = 0$ and $n_1 > 0$

$$.015\ (n_1 + n_3 + 3)$$

d. If $n_1 = 0$ and $n_2 = 0$ (an ISS alone or all spaces and an ISS)

$$.015\ (n_3 + 3)$$

where n_1 = number of digits in multiplicand

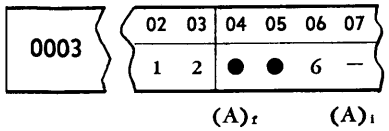
n_2 = number of digits in multiplier

n_3 = total number of spaces (including sign) and/or minuses to right of the LSD's of the operands.

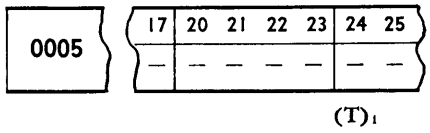
Examples (DM)

1. *Instruction:* 53 000307 00 002311
T Register preset to 000524

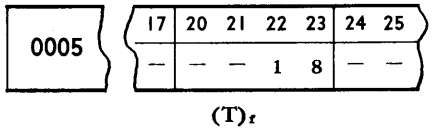
HSM before and after Instruction is executed:



HSM before Instruction is executed:



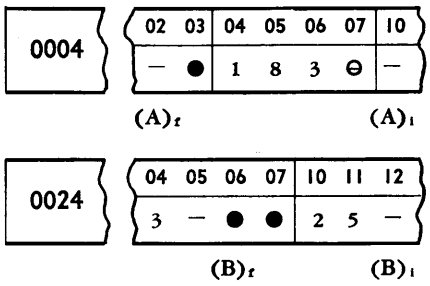
HSM after Instruction:



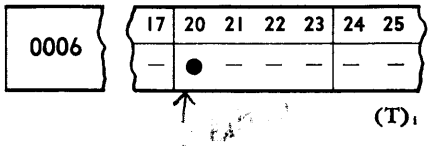
Time:
.015 [10 + (12 x 1 + 32) 1] + (.015 x 2) =
0.840 ms.
PRP is set.

2. *Instruction:* 53 000410 00 002412
T Register preset to 000625

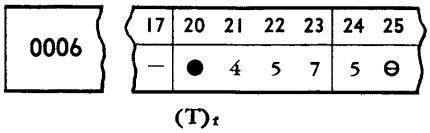
HSM before and after Instruction is executed:



HSM before Instruction is executed:



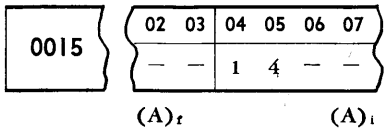
HSM after Instruction:



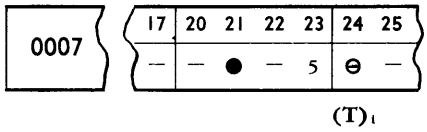
Time:
.015 (10 + [(12 x 3) + 32) 2] + (.015 x 3) =
2.235 ms.
PRN is set.

3. *Instruction:* 53 001507 00 002510
T Register preset to 000724

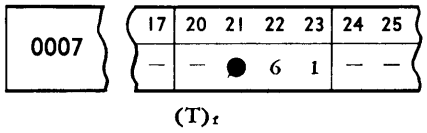
HSM before and after Instruction is executed:



HSM before Instruction is executed:



HSM after Instruction:



Time:
.015 (10 + [(12 x 2) + 32] 2) + (.015 x 2) =
1.860 ms.
PRP is set.

54: Decimal Divide (DD)

General Description

This instruction performs decimal division in accordance with algebraic rules and produces a non-zero-suppressed, non-right-justified quotient. The non-zero-suppressed remainder is stored in the HSM locations originally occupied by the dividend. Each operand must carry a sign. The operands may be of any length. The length of each operand is defined by the first space to the left of a non-space, non-minus character, or by an ISS. If the divisor contains more digits than the dividend, the quotient will be a zero. Unlike Decimal Multiply, the quotient is not added to a prestored quantity.

Format

A address — HSM location of the sign (or space to the right of the sign) of the dividend (and remainder).

B address — HSM location of the sign (or space to the right of the sign) of the divisor.

Preset T Register [see Set Register instruction (72), page 23], so that it specifies the HSM location which is to receive the MSD of the quotient.

Direction of Operation

Shifting and storing of quotient digits is performed from left to right. Individual subtractions are performed from right to left, as in Decimal Subtract (52).

Outline of Logic

In general, division is performed by successively subtracting the divisor from the dividend with the MSD's of the operands aligned. Quotient digits are accumulated according to the number of valid subtractions performed. (An ISS or space is *not* automatically stored to the left of the MSD of the quotient. When a negative remainder is sensed (invalid subtraction), the quotient digit is not increased. The subtractions then begin anew with the divisor shifted to align with the next dividend digit to the right, and a new quotient digit is accumulated.

Initially, a search is made of both operands, during which the following actions are taken:

1. A positive sign is stored in the quotient area if the signs of the operands are alike; a negative sign is stored if they are unlike. Each operand must carry a sign.
2. The divisor is validity checked; an alarm stop occurs if the divisor (a) is missing (i.e., contains only an ISS or spaces and an ISS) or (b) is not zero suppressed.
3. If the divisor contains more digits than the dividend or if the dividend is missing, PRZ is set and a zero with plus sign (space) is placed in the quotient.

Next, the process of subtraction and shifting takes place, with a non-zero-suppressed remainder replacing the dividend after each completed subtraction.

The operation ends when the subtraction process is completed, with the LSD's of the operands aligned.

After the operation the following conditions exist:

1. The non-zero-suppressed remainder occupies all of the HSM locations originally occupied by the dividend. A zero remainder will have a plus sign.
2. The PRI's are set according to the sign of the result. If the divisor is greater in magnitude than the dividend, a single zero and a plus sign will appear in the quotient and PRZ is set.

The number of quotient digits = (number of dividend digits minus number of divisor digits) + 1.

Note: Precision of the quotient may be increased by placing significant zeros in the dividend before the DD instruction is executed.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f$ = HSM location of MSD of remainder minus $(02)_8$.

$(B)_f$ = HSM location of MSD of divisor minus $(02)_8$.

$(T)_f$ = HSM location of the sign of the quotient.

Timing

Total time, in milliseconds:

If $n_1 \geq n_2$

$.015\ [26n_1 - 7n_2 + 15n_2\ (n_1 - n_2) + 41] + .015n_3$

If $n_1 < n_2$

$.015\ (3n_1 + n_2 + 12) + .015n_3$

If $n_1 = 0$ (i.e., the dividend is missing)

$.015\ (n_2 + 7) + .015n_3$

where n_1 = number of digits in the dividend

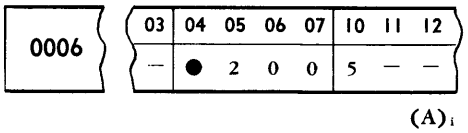
n_2 = number of digits in the divisor

n_3 = total number of spaces (including sign) and/or minuses to the right of the LSD's of the operands.

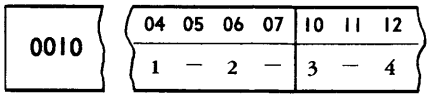
Examples (DD)

1. *Instruction:* 54 000612 00 001011
   T Register preset to 000516

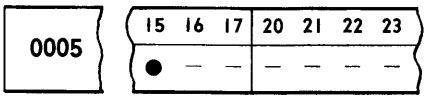
HSM before Instruction is executed:



$(A)_i$

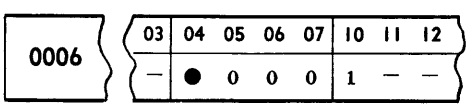


$(B)_i$

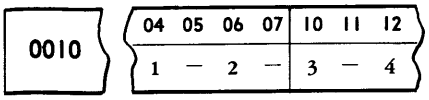


$(T)_i$

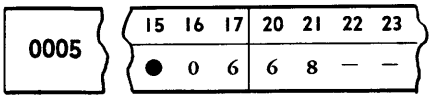
HSM after Instruction:



$(A)_f$



$(B)_t$



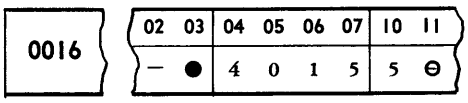
$(T)_t$

Time:

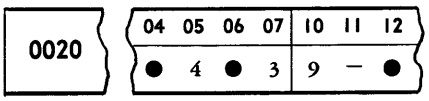
.015 [(26 x 4) − (7 x 1) + 15 (4 − 1) + 41] + (.015 x 3) = 2.790 ms.

2. *Instruction* 54 001611 00 002011
T Register preset to 000416

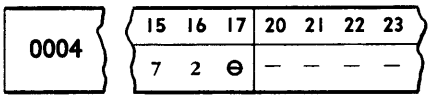
HSM before Instruction is executed:



$(A)_i$

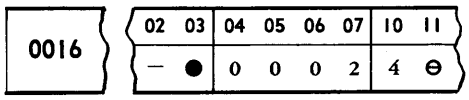


$(B)_i$

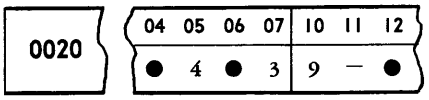


$(T)_i$

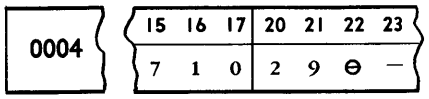
HSM after Instruction:



$(A)_t$



$(B)_t$



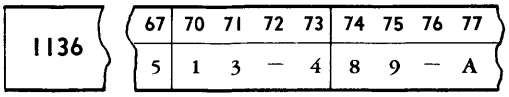
$(T)_t$

Time:

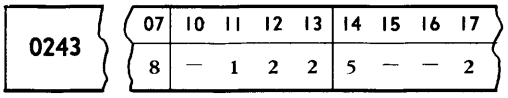
.015 [(26 x 5) − (7 x 2) + 30 (5 − 2) + 41] + (.015 x 2) = 3.735 ms.

3. *Instruction:* 54 113676 00 024315
T Register preset to 027002

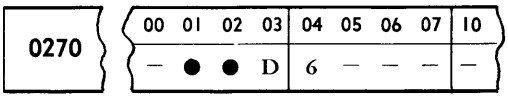
HSM before Instruction is executed:



$(A)_i$

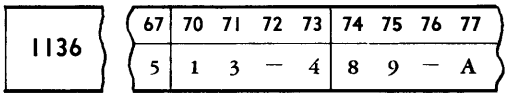


$(B)_i$

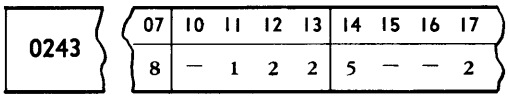


$(T)_i$

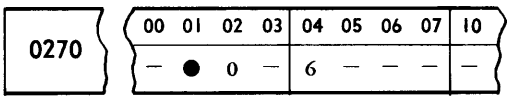
HSM after Instruction:



$(A)_t$



$(B)_t$



$(T)_t$

Time:

.015 [(3 x 3) + 4 + 12] + (.015 x 2) = 0.405 ms.

41: Binary Add (BA)

General Description

This instruction performs binary addition of two equal length operands and places the sum in the HSM locations originally occupied by the augend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

Format

A address — HSM location of the leftmost character of the augend (and sum).

B address — HSM location of the rightmost character of the augend (and sum).

Preset T Register [see Set Register instruction (72), page 23], so that it holds the HSM location of the rightmost character of the addend.

Direction of Operation

Right to left.

Outline of Logic

Binary add is executed according to the following rules:

1. There is no search for the signs or least significant characters of operands and there are no special control symbols. Every character enters into the addition and is treated as if it were numeric.
2. The addition considers all six bits of each character in the operands.
3. Any carry from the most significant bit position of the leftmost character of the sum is discarded.
4. The PRI's are not affected by BA.
5. The rules for binary addition may be stated as follows:

	Sum	*Carry*
$0 + 0 = 0$	0	
$0 + 1 = 1$	0	
$1 + 1 = 0$	1	
$1 + 1 + 1 = 1$	1	

The B Register is used to address the augend and the sum. Immediately before a character is picked up from the augend the contents of the A and B Registers are matched for equality. The BA operation is concluded after A — B equality is attained.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (A)_i - (01)_8$
$(T)_f =$ (MSD of addend) $- (01)_8$

Timing

Total time in microseconds $= 45n$
where n is the number of characters in the augend.

Example (BA)

Instruction: 41 000703 00 000707
T Register is preset to 002155

HSM before Instruction is executed:

0007	00	01	02	03	04	05	06	07
	01	01	01	40	30	20	10	01

$(A)_i$ $(B)_i$

0021	50	51	52	53	54	55	56	57
	01	42	01	74	05	05	05	01

$(T)_i$

HSM after Instruction:

0007	00	01	02	03	04	05	06	07
	01	01	01	02	32	14	15	06

$(B)_f$ $(A)_f$

0021	50	51	52	53	54	55	56	57
	01	42	01	74	05	05	05	01

$(T)_f$

Final register contents:
$(A)_f = 000703$ $(B)_f = 000702$ $(T)_f = 002150$

Time:
$45 \times 5 = 225$ μs.

42: Binary Subtract (BS)

General Description

This instruction performs binary subtraction of one operand from another operand of equal length, placing the difference in the HSM locations originally occupied by the minuend. The operands may be of any length. Each character (including control symbols) is treated as if it were numeric.

Format

A address — HSM location of the leftmost character of the minuend (and difference).
B address — HSM location of the rightmost character of the minuend (and difference).
Preset T Register [see Set Register Instruction (72), page 23], so that it contains the HSM location of the rightmost character of the subtrahend.

Direction of Operation

Right to left.

Outline of Logic

Binary Subtract works according to the following rules:

1. There is no search for signs or the least significant characters of operands and there are no special control symbols. Every character enters into the subtraction and is treated as if it were numeric.
2. The PRI's are set according to the relative magnitude of the operands. However, the sign of the difference is not stored in the HSM.

   PRI settings:
   PRZ if the difference $= (0)_8$.
   PRP if minuend $>$ subtrahend.
   PRN if minuend $<$ subtrahend.
3. All six bits of each character in the operands are considered in the subtraction.

4. The "ones" complement of the subtrahend is taken and a binary addition is performed, with a "one bit" automatically added to the least significant bit position of the difference.
5. Any carry from the most significant bit position of the difference is discarded. If there is a carry, the difference is positive; if no carry, difference is negative.
6. The digits of the minuend are replaced, one by one, with digits of the difference as the operation proceeds.
7. Binary Subtract ends after A — B equality has been attained.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (A)_i - (01)_8$
$(T)_f$ = (MSD of subtrahend) — $(01)_8$.

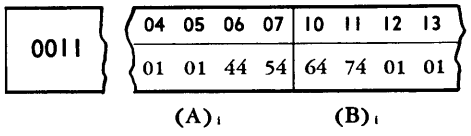
Timing

Total time in microseconds = $45n$
where n is the number of characters in the minuend.

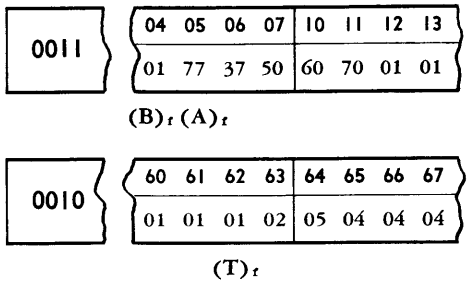
Example (BS)

Instruction: 42 001105 00 001111
T Register is set to 001067

HSM before Instruction:



HSM after Instruction:



Final register contents:

$(A)_f = 001105$ $(B)_f = 001104$ $(T)_f = 001062$
PRN is set.

Time:

45 x 5 = 225 μs.

43: Sector Compare (SC)

General Description

This instruction is used to determine the relative magnitude of two operands of equal length. A single operand may consist of one character or any number of alpha-numeric characters and/or symbols. Binary subtraction is performed, but the difference is not stored in the HSM. However, the resultant PRI settings permit alternative sequences of instructions.

Format

A address — HSM location of the leftmost character of the minuend.
B address — HSM location of the rightmost character of the minuend.
Preset T Register [see Set Register instruction (72), page 23], so that it contains the HSM location of the rightmost character of the subtrahend. (The length of the subtrahend is determined by the length of the minuend.)

Direction of Operation

Right to left.

Outline of Logic

Subtraction is performed exactly as in the Binary Subtract instruction except that the difference is not stored in the HSM.

A positive result sets PRP; a negative result sets PRN; a zero result sets PRZ.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (A)_i - (01)_8$
$(T)_f$ = one HSM location to the left of the leftmost character of subtrahend.

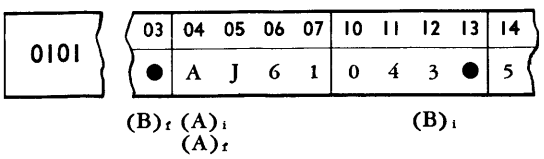
Timing

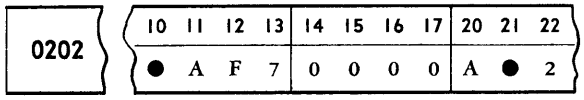
Total time in microseconds = $45n$
where n is the number of characters in the minuend.

Examples (SC)

1. *Instruction:* 43 010104 00 010112
T is preset to 020217

HSM before and after Instruction is executed:



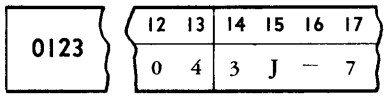


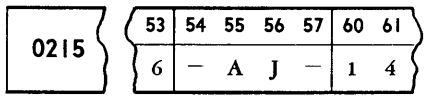
Final register contents:
$(A)_f = 010104$ $(B)_f = 010103$ $(T)_f = 020210$
PRP is set.

Time:
45 x 7 = 315 μs.

2. *Instruction:* 43 012315 00 012315
T is preset to 021556

HSM before and after Instruction is executed:





Final register contents:
$(A)_f = 012315$ $(B)_f = 012314$ $(T)_f = 021555$
PRZ is set.

Time:
45 μs.

44: Three-Character Add (TCA)

General Description

TCA is mainly designed to modify addresses of instructions and to keep octal counters. As such, it performs binary addition of an augend stored in the rightmost three locations of a tetrad and a three-character addend. The result is automatically stored in the locations previously occupied by the augend.

Format

A address — HSM location of rightmost character of the augend (and sum).
B address — HSM location of rightmost character of the addend.

Direction of Operation

Right to left.

Outline of Logic

The TCA instruction operates according to the following rules:

1. There is no search for the least significant characters or signs of the operands. Every character enters into the addition.
2. The PRI's are not affected by the TCA, and a sign is not stored in the sum.
3. The addition considers all six bits of the characters in the operands.
4. Any carry from the most significant bit position of the sum is discarded.
5. All characters are treated as numeric; there are no special control symbols. The addition ends when the two least significant bits in the A Register are 01. (Note that, although this termination condition is constant, addition actually starts with the characters in the locations designated by the A and B addresses. TCA can, therefore, also be used as a one or two-character add if the A address refers to C_1 or C_2 of a tetrad, or as a four-character add if the A address refers to C_0 of the tetrad to the right. This is also applicable to the Three-Character Subtract instruction.)
6. The sum replaces the augend in the HSM, character for character.

Addressable Registers Used

A B

Final Register Contents

$(A)_f$ = (MSD of sum) − $(01)_8$
$(B)_f$ = (MSD of addend) − $(01)_8$

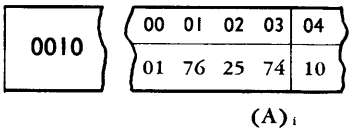
Timing

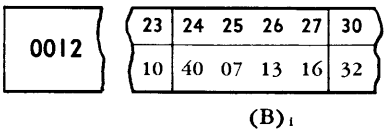
Time in microseconds = $45n$
n = 1, 2, 3 or 4

Example (TCA)

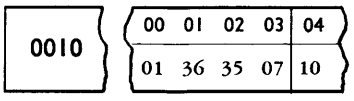
Instruction: 44 001003 00 001226

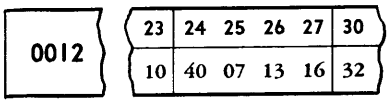
HSM before Instruction is executed:





HSM after Instruction:





Final register contents:

$(A)_f = 001000$ $(B)_f = 001223$

Time:

45 x 3 = 135 μs.

45: Three-Character Subtract (TCS)

General Description

Like the Three-Character Add, TCS is mainly designed for address modification and octal counters. It performs binary subtraction of two operands stored in the rightmost three locations of their respective tetrads. The difference is automatically stored in the locations previously occupied by the minuend.

Format

A address — HSM location of rightmost character of the minuend (and difference).

B address — HSM location of rightmost character of the subtrahend.

Direction of Operation

Right to left.

Outline of Logic

The TCS instruction operates according to the following rules:

1. Every character enters into the subtraction. There is no search for the signs or least significant characters of operands and there are no special control symbols. The operation ends when the two least significant bits in the A Register are 01.
2. A sign is not stored to the right of the difference, although the PRI's are properly set to reflect the relative magnitudes of the operands.
3. The subtraction considers all six bits of characters in the operands.
4. The subtrahend is complemented ("ones" complement) and a binary addition is performed.
5. A "1" (complementary carry) is automatically added into the least significant bit position of the difference. Any carry from the most significant position is discarded. If there was a carry, the difference is positive; if no carry, the difference is negative.
6. The difference replaces the minuend in the HSM.

Addressable Registers Used

A B

Final Register Contents

$(A)_f$ = (MSD of the difference) − $(01)_8$

$(B)_f$ = (MSD of the subtrahend) − $(01)_8$

Timing

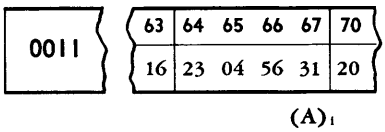
Time in microseconds = $45n$

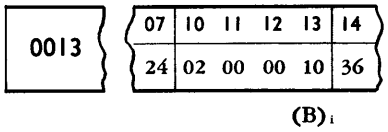
n = 1, 2, 3 or 4

Example (TCS)

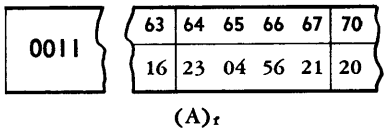
Instruction: 45 001167 00 001313

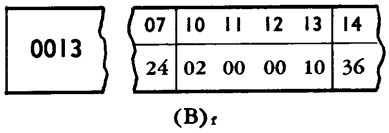
HSM before Instruction is executed:





HSM after Instruction:





Final register contents:

$(A)_f = 001164$ $(B)_f = 001310$

PRP is set.

Time:

45 x 3 = 135 μs.

46: Logical "or" (LO)

General Description

This instruction performs a function similar to the "or" in machine logic, inserting "1" bits from a specified modifier into a specified operand of equal length.

Format

A address — HSM location of the leftmost character of the operand to be modified.

B address — HSM location of the rightmost character of the operand to be modified.

Preset T Register [see Set Register instruction (72), page 23], so that it contains the HSM location of the rightmost character of the modifier.

Direction of Operation

Right to left.

Outline of Logic

LO is a sector instruction; it is completed after A -- B equality is attained.

The operation is performed by a series of bit additions on aligned pairs of characters, proceeding from right to left. All six bits of each character enter the addition, but a carry is not propagated from one bit position to the next. There are no special control symbols.

The rules for the addition are as follows:

$$\begin{cases} 0 + 0 = 0 \\ 1 + 0 = 1 \\ 0 + 1 = 1 \\ 1 + 1 = 1 \end{cases}$$

For example,

$$\begin{array}{r} 011010 \\ +\ 101100 \\ \hline 111110 \end{array}$$

The sum (modified operand) is placed, character by character, in the HSM locations originally occupied by the operand to be modified.

The PRI's are not affected by LO.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (A)_i - (01)_8$

$(T)_f$ = HSM location of leftmost character of the modifier minus $(01)_8$.

Timing

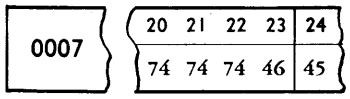
Total time in microseconds = $45n$

where n is the number of characters in the operand to be modified.

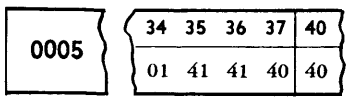
Examples (LO)

1. *Instruction:* 46 000722 00 000724
   T preset to 000540

HSM before Instruction is executed:

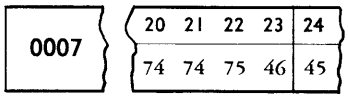


$(A)_i$ $(B)_i$

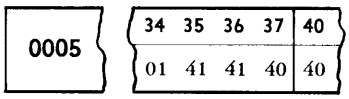


$(T)_i$

HSM after Instruction:



$(B)_f$ $(A)_f$



$(T)_f$

Final register contents:

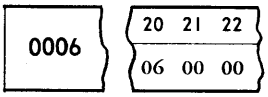
$(A)_f = 000722$ $(B)_f = 000721$ $(T)_f = 000535$

Time:

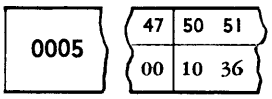
45 x 3 = 135 μs.

2. *Instruction:* 46 000622 00 000622
   Register T preset to 000550

HSM before Instruction is executed:

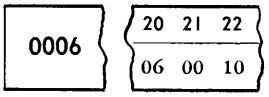


$(A)_i$
$(B)_i$

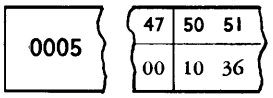


$(T)_i$

HSM after Instruction:



$(B)_f$ $(A)_f$



$(T)_f$

Final register contents:

$(A)_f = 000622$ $(B)_f = 000621$ $(T)_f = 000547$

Time:

45 μs.

47: Logical "and" (LA)

General Description

This instruction performs a function similar to the "and" in machine logic. It may be used to extract "1" bits from an operand according to a second operand ("mask") of equal length.

Format

A address — HSM location of the leftmost character of the operand to be modified.

B address — HSM location of the rightmost character of the operand to be modified.

Preset T Register [see Set Register instruction (72), page 23], so that it contains the HSM location of the rightmost character of the extract pattern (mask).

Direction of Operation

Right to left.

Outline of Logic

LA is a sector instruction; it is completed after A — B equality is attained.

The operation is performed by a series of bit multiplications on aligned pairs of characters, proceeding from right to left. All six bits of each character enter the multiplication, with no carry propagated from one bit position to the next. There are no special control symbols.

The rules for the multiplication are as follows:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

For example,

$$\begin{array}{r} 011010 \\ \times\ 101100 \\ \hline 001000 \end{array}$$

Characters in the operand to be modified are replaced, one by one, by characters of the product.

The PRI's are not affected by this instruction.

Addressable Registers Used

A B T (preset)

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (A)_i - (01)_8$
$(T)_f =$ (MSD of mask) $- (01)_8$

Timing

Total time in microseconds $= 45n$

where n is the number of characters in the operand to be modified.

Example (LA)

Instruction: 47 000722 00 000724
T is set to 000540

HSM before Instruction is executed:

	20	21	22	23	24
0007	74	74	74	46	45
	$(A)_i$			$(B)_i$	

	34	35	36	37	40
0005	01	41	41	40	40
					$(T)_i$

HSM after Instruction:

	20	21	22	23	24
0007	74	74	40	40	40
	$(B)_f$			$(A)_f$	

	34	35	36	37	40
0005	01	41	41	40	40
					$(T)_f$

Final register contents:

$(A)_f = 000722$ $(B)_f = 000721$ $(T)_f = 000535$

Time:

45 x 3 = 135 μs.

71: Transfer Control (TC)

General Description

This instruction either causes an unconditional break in the sequence of instructions or takes action according to the settings of the Breakpoint Switches on the Computer Console.

Format

A address — HSM address of the next instruction to be executed (unless nullified by Breakpoint bits and switch settings).

B address:

B_1 — contains the Breakpoint bits.

B_2B_3 — ignored.

Outline of Logic

The six bits of B_1 are matched against the 6 two-position Breakpoint Switches on the Console. If *any one* of the control bits in B_1 is a "1", and its corresponding Breakpoint Switch is in the "ignore" position, the transfer will not take place and the program will progress to the next instruction in sequence. If the transfer is performed, the contents of the P Register are placed in standard HSM locations and the contents of the A Register are placed in the P Register. In either case, the B address is not placed in the B Register, the Breakpoint bits being examined in either the SW or the SR Register. However, address modification, if called for, will be performed on the contents of the B Register (left by a previous instruction).

Note: STA is not performed by this instruction.

Addressable Registers Used

A

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$ (contents left by a previous instruction)

Timing

15 μs.

61: Conditional Transfer of Control (CTC)

General Description

This instruction chooses one of three sequences of instructions, in accordance with the setting of the PRI's.

Format

A address — HSM location of the next instruction to be executed if PRP is set.

B address — HSM location of the next instruction to be executed if PRN is set.

If PRZ is set, the instruction in the location immediately following the CTC will be executed.

Outline of Logic

The PRI's are examined. If PRZ is set, the next instruction in sequence is staticized. If PRZ is not set, the contents of the P Register (which holds the address of the next instruction in sequence) are placed in standard HSM locations. If PRP is set, the contents of the A Register are transferred to the P Register. If PRN is set, the contents of the B Register are transferred to the P Register.

Note: STA is not performed by this instruction.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (B)_i$

Timing

If PRZ was set, only staticizing time is required.

If PRP or PRN was set, the operation takes 15 μs in addition to staticizing time.

66: Tally (TA)

General Description

This instruction permits looping through a sequence of operations by automatically reducing a prestored quantity each time control is transferred to the beginning of the sequence. When the quantity has been exhausted, the Tally ends and the instruction following it is performed.

Format

A address — HSM location of the next instruction to be performed if the quantity being tested has not been exhausted.

B address — HSM address of the tetrad containing, in its rightmost three locations, the quantity to be tested.

Outline of Logic

The rightmost three characters of the tetrad specified by the contents of the B Register are read out of the HSM and into the T Register. Then, $(000001)_8$ is subtracted from this quantity, and the result is tested. If the result is $(777777)_8$, the Tally ends and the next instruction in sequence is executed. If the result is not $(777777)_8$, it is placed in the HSM locations of the original quantity. The contents of the P Register are then stored in standard HSM locations, and the contents of the A Register are transferred to the P Register. STA is not performed by this instruction.

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (B)_i$

$(T)_f$ = Quantity tested minus $(000001)_8$ each time TA is performed. Note: If the quantity tested is $(000000)_8$, the contents of the T Register will be reduced to $(777777)_8$.

Timing

30 μs if the quantity tested is $(000000)_8$.

60 μs if the quantity is not $(000000)_8$

77: Return After Interrupt (RAI)

General Description

This instruction is used to re-enter a program after an unscheduled interruption, such as for Rollback or for a higher priority program.

The RAI was designed not only to transfer control, but to permit both the A and B Registers to be properly set in the process. Thus, when the main program is re-entered, *all* of the pertinent conditions prevailing at the time of interruption can be re-established.

Format

A address — should contain the address to appear in the A Register when the program is re-entered.

B address — should contain the address to appear in the B Register when the program is re-entered.

Outline of Logic

The RAI instruction automatically transfers into the P Register the contents of standard HSM locations $(000001)_8$ — $(000003)_8$, effecting a transfer of control. Storing of the desired P Register setting into these locations must be accomplished by a previous instruction. Although RAI ignores the A and B addresses, whatever is inserted into these addresses will be transferred into the A and B Registers when the instruction

is staticized, with address modification if indicated, for use on re-entry into the main program. This instruction goes through STA.

Registers Used

A B P

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$
$[(P)_f$ = contents transferred from HSM locations $(000001)_8$ — $(000003)_8$ (i.e., HSM address of next instruction to be executed).]

Timing

15 μs.

76: Stop (ST)

General Description

This instruction inhibits the staticizing of any further instructions, halting the Computer after completion of any instruction in the Simultaneous Mode.

Format

A address — ignored.
B address — ignored.

Outline of Logic

If the Simultaneous Mode is unoccupied when the stop instruction is staticized, the Computer stops immediately. If it is occupied, the Simultaneous instruction is completed before the Computer stops. However, if a "read" parity error is detected in the Simultaneous Mode after a stop instruction is staticized, the Computer will stop, without attempting to perform Rollback.

The stop instruction goes through STA.

Addressable Registers Used

None.

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$

Timing

Staticizing and STA time only.

63: Tape Sense (TS)

General Description

This instruction tests the status of a given Tape Station, permitting program direction to one of two sequences of instructions.

Format

A address — HSM location of the next instruction to be executed if the condition or conditions being tested are present.

B address:

B_1 — specifies the Tape Station number.

B_2 — specifies the tests to be performed. [(111111) in B_2 will perform all six tests; (000001) will perform only the first test.]

"1" Bit in	*Tests*
2^0	Is the tape positioned on BTC?
2^1	Has ETW been sensed?
2^2	Is the tape now stationary or moving forward?
2^3	Is the tape now moving in a reverse direction?
2^4	Is the tape now in motion?
2^5	Is the Tape Station non-operable?

B_3 — Not used.

If the B_1 character is $(77)_8$, the Monitor Printer will be tested if the SR Register is occupied and the Paper Tape Reader will be tested if SR is unoccupied. In either case, the only valid test would be with respect to operability ("1" bit in 2^5).

Outline of Logic

The Tape Station number (B_1) is placed in either the SW or the SR Register (whichever is unoccupied). The tests called for by the "1" bits in B_2 are performed. If *any one* of the conditions tested is present, the contents of the P Register are transferred to standard HSM locations and the contents of the A Register are then transferred to the P Register, effecting transfer of control to the instruction specified by the A Register. If a transfer is not called for, the next instruction in sequence will be executed. The TS instruction does not go through STA.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$

Timing

30 μs if no transfer.
45 μs if a transfer is executed.

02: Print (PR)

General Description

This instruction transfers the characters stored in 120 consecutive HSM locations to the Line Printer, causing one line to be printed. The operation is initiated only

when the Simultaneous Mode is free, since it uses both the Normal and Simultaneous Modes.

Format

A address — leftmost HSM location of the sector to be printed.
A_2 must be an even number.
A_3 must be $(00)_8$.
B address — ignored.

Direction of Operation

Left to right.

Outline of Logic

The contents of the A Register are initially duplicated in the B Register, which is increased by $(04)_8$ after each tetrad processed.

As the drum revolves, the Line Printer sends signals to the Computer, indicating which row will next be in the print position. The sector of 120 characters is read out of the HSM by tetrads and scanned for the character that will next be in print position. Each time the full sector has been scanned, the Line Printer's Shift Register contains 120 "1" and "0" bits, the "1" bits corresponding to the locations in which the character is to be printed. All occurrences of the character are printed at one and the same time. The process is repeated for each of the 54 rows of characters (including the three normally nonprintable characters =, +, and @), so that printing of the 120-character line is completed when the print drum has made one full revolution.

One character is completely printed when the contents of the B Register have been increased by $(170)_8$. [Note: $(170)_8 = (120)_{10}$ = number of characters in the sector.] The operation is terminated when the T Register has been increased to $(003124)_8$. Initially, the T Register is automatically cleared to zeros; $(01)_8$ is added each time a tetrad is scanned [$(120/4)$ $(54) = (1620)_{10} = (3124)_8$].

Addressable Registers Used

A B T (not preset)

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (A)_i + (000170)_8$
$(T)_f = (003124)_8$ = number of tetrads in the sector times the number of character lines around the circumference of the print drum. $(30)_{10} \times (54)_{10} = (1620)_{10} = (3124)_8$.

Timing

One line is printed in 67 milliseconds.

Example (PR)

02 023200 00 000000

This instruction will effect printing, on the Line Printer, of the contents of HSM locations $(023200)_8$ — $(023367)_8$, inclusive.

Final register contents:
$(A)_f = 023200$
$(B)_f = 023370$
$(T)_f = 003124$

Time:
67 ms.

03: Paper Advance (PA)

General Description

This instruction positions the paper in the Line Printer for the next line of printing. It can advance the paper a specified number of lines, designated either by A_2A_3 or by the punches in a tape loop on the Printer. This is a potentially simultaneous instruction. While in the Simultaneous Mode, it does not restrict the use of any other instruction in the Normal Mode except Print or another Paper Advance.

Format

Loop Controlled Advance:
A address:
A_1 — denotes the type of advance.
An odd number (octal) = Vertical Tabulation (VT).
An even number (octal) = Page Change (PC).
A_2A_3 — $(0000)_8$.
B address — ignored.

Instruction Controlled Advance:
A address:
A_1 — ignored.
A_2A_3 — the number (octal count) of lines the paper is to be advanced.
B address — ignored.

Outline of Logic

The A_2A_3 characters are tested. If they are not $(0000)_8$, a signal is sent to the Printer to advance the paper. When one line has been shifted, the Printer signals the Computer and the A_2A_3 characters (in the A or S Register) are decreased by $(0001)_8$. This process continues until $(0000)_8$ emerges from the Bus Adder; the Computer then signals the Printer to stop the Paper Advance. If the A_2A_3 characters are initially $(0000)_8$, the A_1 character is examined, and the function (VT

or PC) specified is performed in accordance with the punches in the tape loop on the Printer. [$(0001)_8$ is subtracted from the A_2A_3 characters (in the A or S Register) with each line shifted.]

This instruction does not use the SW or SR Register, nor is it stored in standard HSM locations.

Addressable Registers Used

A or S

Final Register Contents

If the operation is concluded in the Normal Mode:

$(A_1)_f = (A_1)_i$

$(A_2A_3)_f = (0000)_8$ if $(A_2A_3)_i > (0000)_8$ or

$(A_2A_3)_f = (0000)_8$ minus the number of lines shifted if $(A_2A_3)_i = (0000)_8$

$(B)_f = (B)_i$

If the operation is concluded in the Simultaneous Mode:

$(S)_f$ settings are analogous to $(A)_f$ settings above.

Timing

Total time (paper motion time) in milliseconds:

30 for single line shifting

$30 + (n - 1)\ 20$ for multiline shifting

where n is the number of lines advanced.

Examples (PA)

Loop Controlled:

Instruction:

0	A	N	B
03	050000	00	000000

In this case the paper on the Line Printer will be advanced in accordance with the punches in the VT column of the tape loop on the Printer.

Instruction:

0	A	N	B
03	120000	00	000000

In this case the paper will be advanced in accordance with the punches in the PC column of the tape loop on the Printer.

Computer Controlled:

Instruction:

0	A	N	B
03	000011	00	000000

In this case the paper will be advanced nine lines.

62: Sense Simultaneous Mode (SSM)

General Description

This instruction selects one of four sequences of instructions, depending upon whether the Simultaneous Mode is (1) unoccupied, (2) occupied by a "read" instruction, (3) occupied by a "write" instruction, or (4) occupied by a Paper Advance.

Format

A address — HSM location of the next instruction to be executed if a "read" is in the Simultaneous Mode.

B address — HSM location of the next instruction to be executed if a "write" is in the Simultaneous Mode.

If the Simultaneous Mode is unoccupied, the instruction in the location immediately following the SSM instruction is executed.

If a Paper Advance is in the Simultaneous Mode, control is automatically transferred to HSM location $(000200)_8$. (The programmer will have previously stored, in this location, an instruction that will transfer control to the desired sequence.)

Outline of Logic

A test is made of the Simultaneous Mode indicator. If it is not set, the SSM ends and the Computer continues to the next instruction in sequence. If it is set, the contents of the P Register are stored in standard HSM locations and the "read," "write" and "Paper Advance" indicators are tested, in accordance with which the address of the next instruction to be executed is placed in the P Register.

Note: STA is not performed in this instruction.

Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$

$(B)_f = (B)_i$

Timing

15 μs.

65: Sense Simultaneous Gate (SSG)

General Description

This instruction chooses one of two sequences of instructions, depending upon whether or not the Simultaneous Gate is open.

Format

A address — HSM location of the next instruction to be executed if the Simultaneous Gate is open.

B address — HSM location of the next instruction to be executed if the Simultaneous Gate is closed.

Outline of Logic

The contents of the P Register are stored in standard HSM locations. The gate controlling entrance to the Simultaneous Mode is examined. If it is closed, the contents of the B Register are transferred to the P Register. If the gate is open, the contents of the A Register are transferred to the P Register.

Note: STA is not performed by this instruction.

Addressable Registers Used

A B

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$

Timing

15 μs.

75: Control Simultaneous Gate (CSG)

General Description

This instruction is used to open or close the gate which controls entrance into the Simultaneous Mode, making it possible to either prevent or permit simultaneous operations.

Format

A address — ignored.

B address:

B_1 — must be even (2^0 bit = "0") if gate is to be opened, and odd (2^0 bit = "1") if gate is to be closed.

B_2B_3 — ignored.

Note: This instruction does not go through STA.

Addressable Registers Used

B

Final Register Contents

$(A)_f = (A)_i$
$(B)_f = (B)_i$

Timing

15 μs.

APPENDICES

APPENDIX IV. LIST OF INSTRUCTIONS

INPUT-OUTPUT INSTRUCTIONS

DATA-HANDLING INSTRUCTIONS

ARITHMETIC INSTRUCTIONS

DECISION AND CONTROL INSTRUCTIONS

APPENDIX V. SUMMARY

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.	
00									
01									
02	Print	Leftmost tetrad of sector to be printed; A_3 must be $(00)_8$; A_2 must be even	Ignored	Not preset—used as internal counter	Yes				
03	Paper Advance	A_2A_3 = no. of lines to advance A_1 ignored *Loop-controlled:* $A_2A_3 = (0000)_8$ A_1: 2^0 bit = 1, Vertical Tab. 2^0 bit = 0, Page Change	Ignored	Not preset	Yes			Yes	
04	Linear Read Reverse	Destination tetrad of EM (EM will always be placed in C_3)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes	
05	Block Read Reverse	Destination tetrad of 1st char. transferred from tape (1st char. will always be placed in C_3)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes	
06	Unwind *n* Symbols	A_1 = symbol A_2A_3 = no. of symbols (octal count)	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes	
07									
10									
11	Single Sector Write	HSM loc. of leftmost char. to be written out	HSM loc. of rightmost char.	Preset T_1 = Tape Station* T_2T_3 ignored	Yes			No	
12	Linear Write	HSM loc. of leftmost char. to be written out	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes	
13	Multiple Sector Write	Leftmost tetrad of stored list of addresses	B_1 = Tape Station* B_2B_3 ignored	Not preset—used as internal counter	Yes				

* $(77)_8$ in B_1 will select the Monitor Printer, or Paper Tape Punch via the Monitor Printer, in write instructions, and the Paper Tape Reader in read instructions.

† The contents of the P Register are automatically stored (in standard HSM locations 000241 – 000243) only when control is actually transferred.

	Remarks	Post-Operational Register Settings#		
		A Register	B Register	T Register
	Occupies both modes The character $(00)_8$ must not appear in print-out sector	$(A)_f = (A)_i$	$(B)_f = (A)_i + (000170)_8 = (120)_{10}$ = No. of characters	$(T)_f = (003124)_8 = (30)_{10} \times (54)_{10}$ = 54 cyc/tetrad × 30 tetrads
	A_1 ignored unless $A_2A_3 = (0000)_8$	1. Normal Mode $(A_1)_f = (A_1)_i$ $(A_2A_3)_f = (0000)_8$ if $(A_2A_3)_i > (0000)_8$ or $(A_2A_3)_f = (0000)_8$ minus no. of lines shifted if $(A_2A_3)_i = (0000)_8$ 2. Simultaneous Mode $(S)_f$ settings analogous to $(A)_f$ settings above	1. $(B)_f = (B)_i$	
		1. Normal Mode $(A)_f$ = HSM loc. of SM, ED or EF 2. Simultaneous Mode $(S)_f$ = HSM loc. of SM, ED or EF	1. $(B)_f = (B)_i$	
		1. Normal Mode $(A)_f$ = loc. of last char. read in 2. Simultaneous Mode $(S)_f$ = HSM loc. of last char. read in	1. $(B)_f = (B)_i$	
	A read instr. cannot be simult. with UNS Possible symbols: EM [$(75)_8$], ED [$(73)_8$], EF [$(72)_8$], Gap [$(00)_8$]	1. Normal Mode $(A_1)_f = (A_1)_i$ $(A_2A_3)_f = (0000)_8$ unless PET is reached; then (A_2A_3) = no. of symbols left to be counted 2. Simultaneous Mode $(S_1)_f = (A_1)_i$ $(S_2S_3)_f = (0000)_8$ unless PET is reached; then S_2S_3 = no. of symbols left to be counted	1. $(B)_f = (B)_i$	
		$(A)_f = (B)_i$	$(B)_f = (B)_i$	$(T)_f = (T)_i$
	1st char. is checked for ED or EF (not for SM) LW ends when EM (ED or EF) is written	1. Normal Mode $(A)_f$ = HSM loc. of EM, EF or ED 2. Simultaneous Mode $(S)_f$ = HSM loc. of EM, EF or ED	1. $(B)_f = (B)_i$	
	MSW ends when XXX in stored list is $(000000)_8$	$(A)_f$ = Address of last tetrad in stored list of addresses	$(B)_f = (B)_i$	$(T)_f = (000000)_8$

‡ If the B_1 character is $(77)_8$, the Monitor Printer will be tested if the SR Register is occupied and the Paper Tape Reader will be tested if the SR Register is unoccupied. In either case, the only valid test would be with respect to operability ("1" bit in 2^5).

Register settings will reflect automatic address modification.

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
14	Linear Read Forward	Destination tetrad of SM (ED or EF) (SM, ED or EF will be placed in C_0)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
15	Block Read Forward	Destination tetrad of 1st char. read (this char. will be placed in C_0)	B_1 = Tape Station* B_2B_3 ignored	Not preset	Yes			Yes
16	Rewind *n* Symbols	A_1 = Symbol to be counted A_2A_3 = No. of symbols to be counted	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes
17	Rewind to BTC	Ignored	B_1 = Tape Station B_2B_3 ignored	Not preset	Yes			Yes
20								
21	Item Transfer	HSM loc. of rightmost char. to be transferred	Destination loc. of rightmost char.	Not preset	Yes		Yes	
22	One-Character Transfer	HSM loc. of char. to be transferred	Destination loc. of char.	Not preset	Yes			
23								
24	Sector Transfer by Character	HSM loc. of leftmost char. of sector to be transferred	HSM loc. of rightmost char. of sector to be transferred	Preset—destination loc. of rightmost char.	Yes			
25	Three-Character Transfer	Address of the tetrad containing, in C_1, C_2 and C_3, the characters to be transferred	Address of the tetrad to receive, in C_1, C_2 and C_3, the three characters.	Not preset	Yes			
26	Sector Transfer by Tetrad	Leftmost tetrad of sector to be transferred	Rightmost tetrad of sector to be transferred	Preset—destination (tetrad address) of rightmost tetrad	Yes			
27	Random Distribute	HSM loc. of SM or ISS of leftmost item in sector to be distributed	Address of the tetrad in the stored list which contains, (in C_1, C_2 and C_3), the destination loc. of the SM or the ISS of the leftmost item to be distributed	Not preset—used for internal addressing	Yes		Yes	

See footnotes on pages 64 and 65.

INSTRUCTIONS (Continued)

	Remarks	Post-Operational Register Settings#		
		A Register	B Register	T Register
		1. Normal Mode $(A)_f = $ HSM loc. of EM 2. Simultaneous Mode $(S)_f = $ HSM loc. of EM	1. $(B)_f = (B)_i$	
		1. Normal Mode $(A)_f = $ HSM loc. of last char. read in 2. Simultaneous Mode $(S)_f = $ HSM loc. of last char. read in	1. $(B)_f = (B)_i$	
	A read instr. cannot be simult. with RNS Possible symbols: SM $[(76)_8]$, ED $[(73)_8]$, EF $[(72)_8]$, Gap $[(00)_8]$	1. Normal Mode $(A_1)_f = (A_1)$ $(A_2A_3)_f = (0000)_8$ unless the BTC was reached; then $(A_2A_3)_f = $ no. of symbols left to count 2. Simultaneous Mode $(S_1)_f = (S_1)_i$ $(S_2S_3)_f = (0000)_8$ unless the BTC was reached; then $(S_2S_3)_f = $ no. of symbols left to count	1. $(B)_f = (B)_i$	
	Rewind is completely free of Computer after start	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	IT ends on transfer of ISS If item contains one or more non-space characters (to the right of the ISS), PRP is set; if only space symbols, PRN is set	Original HSM loc. of ISS minus $(01)_8$	Destination loc. of ISS minus $(01)_8$	
		$(A)_f = (A)_i$	$(B)_f = (B)_i$	
		$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	$(T)_f = (T)_i - n$ $n = $ no. of char. transferred
	Characters are transferred in parallel	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
		$(A)_f = (A)_i$	$(B)_f = (B)_i - 4n$ $n = $ no. of tetrads transferred	$(T)_f = (T)_i - 4n$ $n = $ no. of tetrads transferred
	1. If terminal address in list has been sensed and the EM has not been sensed, PRZ is set 2. If terminal address in list has been sensed and EM has been sensed, PRP is set 3. If terminal address in list has not been sensed and EM has been sensed, PRN is set EM is not distributed; ISS placed in destination loc. instead	HSM loc. of last control symbol (ISS, SM or EM) sensed in original area	$(B)_f = (B)_i + 4n$ $n = $ no. of addresses in stored list (including terminal and "throw-away" addresses)	$(T)_f = (000000)_8$

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.	
30									
31	Locate *n*th Symbol in Sector	Leftmost HSM loc. of sector to be searched	Rightmost HSM loc. of sector to be searched	Preset T_1 = Symbol to be counted T_2T_3 = No. of symbols to be counted	Yes		Yes		
32	Zero Suppress	Leftmost HSM loc. of sector in which zeros are to be suppressed	Rightmost HSM loc. of sector in which zeros are to be suppressed	Not preset	Yes				
33	Justify Right	Rightmost HSM loc. of item to be justified	Destination loc. of sign or LSC of item	Not preset—used as internal counter	Yes		Yes		
34	Sector Clear by Character	Leftmost HSM loc. of sector to be cleared	Rightmost HSM loc. of sector to be cleared	Not preset	Yes				
35	Sector Compress—Retain Redundant ISS's	Leftmost HSM loc. of sector to be compressed	Rightmost HSM loc. of sector to be compressed	Preset—destination loc. of rightmost retained char.	Yes		Yes		
36	Sector Clear by Tetrad	Address of leftmost tetrad to be cleared	Address of rightmost tetrad to be cleared	Not preset	Yes				
37	Sector Compress—Delete Redundant ISS's	Leftmost HSM loc. of sector to be compressed	Rightmost HSM loc. of sector to be compressed	Preset—destination loc. of rightmost retained char.	Yes		Yes		
40									
41	Binary Add	HSM loc. of leftmost char. of augend (and sum)	HSM loc. of rightmost char. of augend (and sum)	Preset—HSM loc. of rightmost char. of addend	Yes				
42	Binary Subtract	HSM loc. of leftmost char. of minuend (and difference)	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Yes		Yes		
43	Sector Compare	HSM loc. of leftmost char. of minuend	HSM loc. of rightmost char. of minuend	HSM loc. of rightmost char. of subtrahend	Yes		Yes		

See footnotes on pages 64 and 65.

INSTRUCTIONS (Continued)

	Remarks	Post-Operational Register Settings#		
		A Register	B Register	T Register
	PRZ set if $(T_2T_3)_f = (0000)_8$ and A $\neq$ B PRP set if $(T_2T_3)_f = (0000)_8$ and A = B PRN set if $(T_2T_3)_f \neq (0000)_8$ and A = B Instruction not performed, but PRI's set: if $(T_2T_3)_i = (0000)_8$ (PRZ) if $(A)_i = (B)_i$ (PRN)	1. If LNS concluded with $(T_2T_3)_f = (0000)_8$: $(A)_f = (A)_i + n - (02)_8$ n = no. of symbols counted 2. If LNS concluded with A = B $(A)_f = (B)_i - (01)_8$	$(B)_f = (B)_i$	1. $(T_2T_3)_f = (0000)_8$ 2. $(T_2T_3)_f = (T_2T_3)_i$ minus no. of symbols counted
		1. $(A)_f = (A)_i + n - (02)_8$ n = number of locations searched 2. If ZS terminated by A-B equality $(A)_f = (B)_i$	$(B)_f = (B)_i$	1. $(T)_f = (A)_f + (01)_8$ 2. If ZS terminated by A-B equality (a) Only zeros encountered $(T)_f = (B)_i + (01)_8$ (b) Only spaces encountered $(T)_f = (B)_i$
	If a non-space char. is encountered prior to the ISS, PRP is set (minus sign = non-space char.) If only an ISS or spaces and ISS, PRN is set	Original loc. of ISS minus $(01)_8$	Destination loc. of ISS minus $(01)_8$	$(000000)_8$ if 1st char. in original loc. is space or minus $(777777)_8$ if 1st char. in original loc. is not space or minus
		$(A)_f = (A)_i$	$(B)_f = (B)_i - n = (A)_i - (01)_8$ n = no. of char. cleared	
	If the sector contains any non-space, non-EM, non-ISS char., PRP is set If the sector contains only spaces symbols, EM, and/or ISS, PRN is set	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	$(T)_f = (T)_i - n$ n = no. of char. transferred
		$(A)_f = (A)_i$	$(B)_f = (B)_i - 4n$ n = no. of tetrads cleared	
	If sector contains any non-space, non-ISS char., PRP is set If sector contains only space symbols and/or ISS, PRN is set Redundant ISS's not deleted if $(B)_i$ = loc. of EM or loc. to right of EM	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	$(T)_f = (T)_i - n$ n = no. of char. actually transferred
	All characters treated as numeric; no carry from most signif. bit position of sum	MSC of sum	$(B)_f = (A)_f - (01)_8$	MSC of addend minus $(01)_8$
	No carry from most signif. bit position of difference; sign of difference not stored in HSM; PRZ set if diff. = octal zero; PRP set if minuend > subtrahend; PRN set if minuend < subtrahend	HSM loc. of MSC of difference	$(B)_f = (A)_f - (01)_8$	HSM loc. of MSC of subtrahend minus $(01)_8$
	PRI settings same as for Binary Subtract Difference not stored in HSM	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	HSM loc. of MSC of subtrahend minus $(01)_8$

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.
44	Three-Character Add	HSM loc. of rightmost char. of augend (and sum)	HSM loc. of rightmost char. of addend	Not preset	Yes			
45	Three-Character Subtract	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Not preset	Yes		Yes	
46	Logical "or"	HSM loc. of leftmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of modifier	Yes			
47	Logical "and"	HSM loc. of leftmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of operand to be modified (and result)	HSM loc. of rightmost char. of modifier (mask)	Yes			
50								
51	Decimal Add	HSM loc. of rightmost char. of augend (and sum)	HSM loc. of rightmost char. of addend	Not preset—used internally to address the sum	Yes		Yes	
52	Decimal Subtract	HSM loc. of rightmost char. of minuend (and difference)	HSM loc. of rightmost char. of subtrahend	Not preset—used internally to address the difference	Yes		Yes	
53	Decimal Multiply	HSM loc. of rightmost char. of multiplicand	HSM loc. of rightmost char. of multiplier	Preset—destination loc. of sign of product	Yes		Yes	
54	Decimal Divide	HSM loc. of rightmost char. of dividend (and remainder); must be sign or space to right of sign	HSM loc. of rightmost char. of divisor; must be sign or space to right of sign	Preset—HSM loc. of leftmost digit of quotient	Yes		Yes	
55								
56								
57								
60								
61	Conditional Transfer of Control	Address of next instr. if PRP is set	Address of next instr. if PRN is set	Not preset		Yes		
62	Sense Simultaneous Mode	Address of next instr. if a "read" is in Simult. Mode.	Address of next instr. if a "write" is in Simult. Mode			Yes		
63	Tape Sense	Address of next instr. if any of the tested conditions are present	Preset B_1 = Tape Station‡ B_2 = Tests to be performed B_3 ignored	Not preset		Yes		
64								
65	Sense Simultaneous Gate	Address of next instr. if Simult. Gate is open	Address of next instr. if Simult. Gate is closed	Not preset		Yes		
66	Tally	Address of next instr. if Tally quantity is not $(000000)_8$	Address of tetrad containing (in C_1, C_2 and C_3) the Tally quantity	Not preset—used as internal counter		Yes		

See footnotes on pages 64 and 65.

INSTRUCTIONS (Continued)

	Remarks	Post-Operational Register Settings#		
		A Register	B Register	T Register
	TCA ends on a C_1 char. See also Binary Add (Remarks)	HSM loc. of MSC of sum minus $(01)_8$	HSM loc. of MSC of addend minus $(01)_8$	
	TCS ends on a C_1 char. See also Binary Subtract (Remarks)	HSM loc. of MSC of difference minus $(01)_8$	HSM loc. of MSC of subtrahend	
	Control symbols treated as data	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	HSM loc. of MSC of modifier minus $(01)_8$
	Control symbols treated as data	$(A)_f = (A)_i$	$(B)_f = (A)_i - (01)_8$	HSM loc. of MSC of modifier minus $(01)_8$
	A Register must initially address a space, minus or ISS	HSM loc. that held the MSD of augend minus $(02)_8$	HSM loc. of MSD of addend minus $(02)_8$	HSM loc. of ISS of sum
	A Register must initially address a space, minus or ISS	HSM loc. that held the MSD of minuend minus $(02)_8$	HSM loc. of MSD of subtrahend minus $(02)_8$	HSM loc. of ISS of difference
	PRI settings are related to the product (not the accumulated result)	HSM loc. of MSD of multiplicand minus $(02)_8$	HSM loc. of MSD of multiplier minus $(02)_8$	HSM loc. of MSD of product minus $(01)_8$
	No. of quotient digits =(No. of divisor digits) + 1 PRZ set if divisor > dividend PRP set if signs of operands alike PRN set if signs of operands unlike	HSM loc. of MSD of remainder minus $(02)_8$	HSM loc. of MSD of divisor minus $(02)_8$	HSM loc. of sign of quotient
	Refers to, but does not set, PRI's; if PRZ set, takes next instr. in sequence, and no STP	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	Control transferred to 000200 if PA in Sim. Mode; if Sim. Mode unoccupied, next instr. in sequence and no STP	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	See instr. write-up for conditions tested by B_2 bits Next instr. in sequence and no STP if none of the conditions tested is present	$(A)_f = (A)_i$	$(B)_f = (B)_i$	
		$(A)_f = (A)_i$	$(B)_f = (B)_i$	
	STP not performed if quantity tested is $(000000)_8$	$(A)_f = (A)_i$	$(B)_f = (B)_i$	$(T)_f$ = quantity tested minus $(000001)_8$; if $(T)_i = (000000)_8$, $(T)_f = (777777)_8$

Op. Code	Instruction	A Address	B Address	T Register	STA	STP†	Sets PRI's	P.S.	
67									
70									
71	Transfer Control	Address of next instr. to be executed	B_1 = Breakpoint bits B_2B_3 ignored	Not preset		Yes			
72	Set Register	Actual value to be placed in the register specified by B_1	B_1 = Register to be set B_2B_3 ignored	Not preset					
73	Store Register	Address of the tetrad to receive (in C_1, C_2 and C_3) contents of register specified by B_1	B_1 = Register whose contents are to be stored B_2B_3 ignored	Not preset					
74									
75	Control Simultaneous Gate	Ignored	B_1 = Even = Open Gate Odd = Close Gate B_2B_3 ignored	Not preset					
76	Stop	Ignored	Ignored	Not preset	Yes				
77	Return After Interrupt	Actual address to appear in A Reg. when program is re-entered	Actual address to appear in B Reg. when program is re-entered	Not preset	Yes				

See footnotes on pages 64 and 65.

INSTRUCTIONS (Continued)

	Remarks	Post-Operational Register Settings#		
		A Register	B Register	T Register
	TC alterable by Breakpoint switch settings	$(A)_f = (A)_i$	TC does not use B Register $(B)_f = (B)_i$ = contents left by previous instr.	
	Can set A, P, T and PRI's	$(A)_f = (A)_i$	SET does not use B Register $(B)_f = (B)_i$ = contents left by previous instr.	
	Can store B, P, S, T and PRI's	$(A)_f = (A)_i$	STR does not use B Register $(B)_f = (B)_i$ = contents left by previous instr.	
		$(A)_f = (A)_i$	$(B)_f = (B)_i$	
		$(A)_f = (A)_i$	$(B)_f = (B)_i$	
		$(A)_f = (A)_i$	$(B)_f = (B)_i$	

APPENDIX VI. INSTRUCTION TIMING*

Op Code	Instruction	Timing in μs	Expository Notes
02	Print†	67	One 120-character line
03	Paper Advance†	a. 30 b. 30 + (n–1) 20	a. Single line shifting b. Multiline shifting n = no. of lines skipped
04	Linear Read Reverse†‡	3.575 + .03n	n = no. of characters read
05	Block Read Reverse†‡	3.575 + .03 (n + 6)	n = no. of characters read
06	Unwind n Symbols†	3.575 + .03n + 4m	n = total no. of characters read, including symbols being counted m = no. of gaps encountered
11	Single Sector Write†‡	3.575 + .03n	n = no. of characters read
12	Linear Write†‡	3.575 + .03n	n = no. of characters read
13	Multiple Sector Write†‡	3.575 + .03n + .03 (m–1)	n = no. of characters read m = total no. of addresses in stored list
14	Linear Read Forward†‡	3.575 + .03n	n = no. of characters read
15	Block Read Forward†‡	3.575 + .03 (n + 6)	n = no. of characters read
16	Rewind n Symbols	3.575 + .03n + 4m	n = total no. of characters read, including symbols being counted m = no. of gaps encountered
17	Rewind to BTC	a. 300 b. 105	a. If the BTC is not positioned at the read-write head when the RWD instruction is given b. If the BTC is already positioned at the read-write head when the RWD instruction is given
21	Item Transfer	30n	n = no. of characters transferred
22	One-Character Transfer	30	
24	Sector Transfer by Character	30n	n = no. of characters transferred
25	Three-Character Transfer	30	
26	Sector Transfer by Tetrad	30n	n = no. of tetrads transferred
27	Random Distribute#	$33n_1 + 18n_2 + 45n_3$	n_1 = total no. of characters transferred n_2 = total no. of characters whose distribution address is $(777777)_8$ n_3 = no. of distribution addresses left when EM is found (the address of the EM must be included in n_3)
31	Locate nth Symbol in Sector	15m + 30n + 45	n = no. of occurrences counted m = total no. of locations searched
32	Zero Suppress	a. 15m + 30n + 15 b. 15m + 30n c. 30	a. In the usual case b. If ZS terminated by A-B equality c. If no zeros or spaces found m = no. of spaces preceding 1st non-space character n = no. of zeros suppressed
33	Justify Right	30n + 30m	n = no. of space and/or minus characters to the right of the rightmost non-minus, non-space character m = no. of non-space, non-minus characters (including ISS) transferred
34	Sector Clear by Character	15n	n = total no. of locations cleared

* STA and staticizing time are not included in the formulas.

† Time here is in milliseconds.

‡ Magnetic tape time. Time for the Monitor Printer is 10 char./sec.; for the Paper Tape Reader, 400 char./sec. Start time for Monitor Print and Paper Tape Read is negligible and need not be computed.

Timing formula is weighted average.

APPENDIX VI. INSTRUCTION TIMING* (Continued)

Op Code	Instruction	Timing in μs	Expository Notes
35	Sector Compress — Retain Redundant ISS's	$15n + 15m$	m = total no. of characters in original sector n = total no. of characters actually transferred
36	Sector Clear by Tetrad	$15n$	n = no. of tetrads cleared
37	Sector Compress — Delete Redundant ISS's	$15n + 15m$	m = total no. of characters in original sector n = total no. of characters transferred
41	Binary Add	$45n$	n = no. of characters in augend
42	Binary Subtract	$45n$	n = no. of characters in minuend
43	Sector Compare	$45n$	n = no. of characters in minuend
44	Three-Character Add	$45n$	n = 1, 2, 3, or 4
45	Three-Character Subtract	$45n$	n = 1, 2, 3, or 4
46	Logical "or"	$45n$	n = no. of characters in operand to be modified
47	Logical "and"	$45n$	n = no. of characters in operand to be modified
51	Decimal Add	$15n_1 + 45n_2 + 30n_3 + 90$	n_1 = total no. of space and/or minus characters found to the right of both operands n_2 = no. of digits in the shorter operand n_3 = difference in no. of digits of the two operands If neg. result, add $30(n + 1) + 15$ n = no. of digits in sum
52	Decimal Subtract	Same as Decimal Add	
53	Decimal Multiply†	a. $n_1 > 0$ and $n_2 > 0$ $.015\ [10 + (12n_1 + 32)\ n_2] + .015n_3$ b. $n_1 = 0$ and $n_2 > 0$ $.015\ (n_2 + n_3 + 3)$ c. $n_2 = 0$ and $n_1 > 0$ $.015\ (n_1 + n_3 + 3)$ d. $n_1 = 0$ and $n_2 = 0$ $.015\ (n_3 + 3)$	n_1 = no. of digits in multiplicand n_2 = no. of digits in multiplier n_3 = total no. of spaces (including sign) and/or minuses to right of LSD's of operands
54	Decimal Divide†	a. $n_1 \geq n_2$ $.015\ [26n_1 - 7n_2 + 15n_2\ (n_1 - n_2) + 41] + .015n_3$ b. $n_1 < n_2$ $.015\ (3n_1 + n_2 + 12) + .015n_3$ c. $n_1 = 0$ (dividend missing, i.e., an ISS alone or all spaces and an ISS) $.015\ (n_2 + 7) + .015n_3$	n_1 = no. of digits in dividend n_2 = no. of digits in divisor n_3 = total no. of spaces (including sign) and/or minuses to the right of LSD's of operands
61	Conditional Transfer of Control	a. Staticizing time only b. 15	a. If zero path taken b. If plus or minus path taken
62	Sense Simultaneous Mode	15	
63	Tape Sense	a. 30 b. 45	a. If no transfer of control b. If transfer executed
65	Sense Simultaneous Gate	15	
66	Tally	a. 30 b. 60	a. If quantity tested is $(000000)_8$ b. If quantity greater than $(000000)_8$
71	Transfer Control	15	
72	Set Register	a. 15 b. 30	a. Set P, T, or PRI's b. Set A Register
73	Store Register	15	
75	Control Simultaneous Gate	15	
76	Stop	Stat. & STA time only	
77	Return After Interrupt	15	

See footnotes on page 74.

STANDARD HIGH-SPEED MEMORY LOCATIONS

	00 01 02 03	04 05 06 07	10 11 12 13	14 15 16 17	20 21 22 23	24 25 26 27	30 31 32 33	34 35 36 37	40 41 42 43	44 45 46 47	50 51 52 53	54 55 56 57	60 61 62 63	64 65 66 67	70 71 72 73	74 75 76 77	
00	RAI	ARITHMETIC TEMPORARY STORAGE			READ INSTRUCTIONS		WRITE INSTRUCTIONS		ROLLBACK ENTRY NORMAL MODE		ROLLBACK ENTRY SIMULT. MODE		PLAN ROLLBACK IN PART II MODE		FLAW MAP ENTRY IN PART II MODE		00
01			A.M. 1				A.M. 3				A.M. 5				A.M. 7		01
02	Should Contain Next Instr. If PA in Simult. Mode				STA A.M. 2				STP								02
03																	03

TITLE:_____ BLOCK NO.:_____ INDEX NO:_____ PROGRAMMER:_____ DATE:_____ PAGE_____ OF_____

RCA 154 5/58

APPENDIX VII. STANDARD HIGH-SPEED MEMORY LOCATIONS AND LIST OF ADDRESS MODIFIERS

HSM Locations	*Use*
000001 – 000003	Return After Interrupt. The RAI instruction effects a transfer of control to the instruction address stored in these locations.
000004 – 000007 000010 – 000013 000014 – 000017	Utilized by arithmetic instructions, for temporary storage of addresses, in lieu of special registers. (The contents of these locations are not useful to the programmer.)
000020 – 000027	Storage locations for read (including unwind and rewind) instructions after staticizing and address modification.
000030 – 000037	Storage locations for write instructions after staticizing and address modification. If a read or write instruction contains an N character other than $(00)_8$, the instruction will be stored with the A and/or B address modified and the N character changed to $(00)_8$.
000040 – 000047	Rollback entrance—Normal. Control transferred to 000040 if the instruction in which the error occurred was in the Normal Mode.
000050 – 000057	Rollback entrance—Simultaneous. Control transferred to 000050 if the instruction in which the error occurred was in the Simultaneous Mode.
000111 – 000113 (A.M. 1) 000131 – 000133 (A.M. 3) 000151 – 000153 (A.M. 5) 000171 – 000173 (A.M. 7)	Static Address Modifiers.
000200	Control transferred to this address if a Paper Advance is sensed (Sense Simultaneous Mode instruction) in the Simultaneous Mode.
000221 – 000223	STA and A.M. 2.
000241 – 000243	STP

ADDRESS MODIFIERS

*Octal Digit**	*Location of Modifier*
0	No Modifier
1	HSM locations 000111 – 000113
2	HSM locations 000221 – 000223 (STA)
3	HSM locations 000131 – 000133
4	P Register
5	HSM locations 000151 – 000153
6	T Register
7	HSM locations 000171 – 000173

* Either digit of the N character of an instruction.

APPENDIX VIII. GLOSSARY

Access Time. A time interval which is characteristic of a storage device, and is essentially a measure of the time required to communicate with that device.* The time interval between (1) the instant at which information is called for from storage and the instant at which it is delivered or (2) the instant which information is ready for storage and the instant at which it is stored.

Accuracy. The quality of freedom from mistake or error, that is, of conformity to truth or to a rule. Accuracy is distinguished from precision as in the following example: A six-place table is more precise than a four-place table. However, if there are errors in the six-place table, it may be either more or less accurate than the four-place table.*

Address (noun). An expression, usually numerical, which designates a particular location in a storage or memory device or other source or destination of information.*

Absolute Address (Actual Address). The specific label assigned by the machine designer to a particular storage location. To code in absolute means to write a sequence of instructions in a computer code.

Instruction Address. (Line Number, Location). An expression used in coding to denote the address of a stored instruction. NOT a part of the instruction itself.

Symbolic Address. A label expressed in a pseudo-code. To code using symbolic addresses implies that the sequence of instructions must be translated into absolute before being executed by a computer. Relative addresses are those symbolic addresses which are translated into absolute by sequencing from some specific "reference" address.

Address Modifier. See text, page 14, and Appendix VII.

Batch. Several groups of items in sequence, each separated by a special symbol and the entire grouping bracketed by SM–EM. (This is contrasted to a single group of related items for a message.)

Beginning of Tape Control (BTC). A "window" placed at the beginning of a magnetic tape, where recording of data is not possible and which can be sensed photoelectrically.

Binary Digit. See Digit.

Binary Representation. See Positional Notation.

Bit. Contraction of Binary Digit.

Block. See page 6.

Breakpoint Switch. There are 6 two-positional breakpoint switches on the Console in the RCA 501 System. When one of these is in the "ignore" position and the related control bit is present in B_1 of a Transfer Control (71) instruction, the transfer will not take place; instead, the program will proceed to the next instruction in sequence.

Buffer. A storage device used to compensate for a difference in rate of flow of information or in time of occurrence of events when transmitting information from one device to another, as from an input device to the High-Speed Memory, or from the High-Speed Memory to an output device.

Carry. (1) A condition occurring during addition when the sum of two digits in the same column equals or exceeds the base number. (2) The digit to be added to the next higher column.

Character. One of a set of elementary symbols which may be arranged in ordered aggregates to express information. These symbols may include decimal digits 0 through 9, the letters A through Z, punctuation symbols, typewriter symbols, and any other symbols which a computer may read, store, or write. See Appendix II for list of RCA characters.

Code (noun). A system of symbols and rules for their use in representing information. A language.

Pulse Code. The binary representation of characters.

Operation Code. The code representing an operation (add, subtract, transfer, etc.) built into the hardware of the computer.

Complement (noun). A quantity which is derived from a given computer quantity by the following rules.

a. Complement on n (as in tens complement). Subtract each digit of the given quantity from n-1, add unity to the least significant digit, and perform all resultant carries.

b. Complement on n-1 (as in nines complement). Subtract each digit of the given quantity from n-1.

Constant. A number is said to be a constant if it has the same value under all conditions. For example, in the formula (area of a circle) = Pi $\times$ (radius)^2, Pi is a constant, equal to 3.14159– – –, which applies to all circles.

Control Symbol. A character used to indicate the beginning or the end of a unit of data *(item, message, file, etc.).*

Counter. A device (register or storage location) for storing integers, permitting these integers to be

* Institute of Radio Engineers definition.

increased or decreased by unity or by an arbitrary integer, and capable of being reset to zero or to an arbitrary integer.

Criterion (Key). A group of characters, usually comprising an *item,* used to identify a *message.*

Decimal Number System. See Positional Notation.

Digit. One of the *n* symbols of integral value ranging from 0 to *n*-1, inclusive, in a scale of numbering of base *n.*

Binary Digits are 0 and 1.

Octal Digits are 0 through 7.

Decimal Digits are 0 through 9.

Edit. To rearrange information. Editing may involve the deletion of unwanted data, the selection of pertinent data, the insertion of invariant symbols such as page numbers and typewriter characters, and the application of standard processes such as zero suppression.

End Data Symbol. See Organization of Data on Tape (text).

End File Symbol. See Organization of Data on Tape (text).

End Message Symbol. See Organization of Data on Tape (text).

End of Tape Warning (ETW). A warning generated by a metal strip located 15 to 20 feet before the physical end of tape.

File. See Organization of Data on Tape (text).

Flip-Flop. A device having two stable states and two input terminals (or types of input signals), each of which corresponds to one of the two states. (The two states may be considered as corresponding to "off" and "on," or to binary 0 and 1. The circuit remains in either state until it is caused to change to the other state by application of the corresponding signal.*

Gap. See Organization of Data on Tape (text).

High-Speed Memory. Magnetic core storage in the Computer in the RCA 501 System. See also Storage.

High-Speed Memory (HSM) *Location.* A unit of magnetic core storage (High-Speed Memory) which can store (hold, remember) one RCA character (one octal number, two octal digits).

Input. (1) Information transferred into the computer. (2) The device by means of which information is fed into the computer.*

Instruction. A set of symbols which directs the computer to take a given action.

Intermessage Gap. See Organization of Data on Tape (text).

Item. See Organization of Data on Tape (text).

Item Separator Symbol (ISS). Control symbol designating the beginning of an item.

Jump Table. Record indicating executed transfers out of program sequence.

Justify. Shift an operand to effect right or left columnar alignment.

Key. See Criterion.

Line. See Organization of Data on Tape (text).

Location. See High-Speed Memory (HSM) Location; Address; Storage.

Mask. A pattern consisting of 0 and/or 1 bits, used to alter the bit configuration of an operand.

Memory. See Storage.

Message. See Organization of Data on Tape (text).

Number System. See Positional Notation.

Octal. See Positional Notation.

Octonary. See Positional Notation.

Operand. Any one of the quantities entering into an operation.*

Output (noun). (1) Information transferred from the computer to external storage.* (2) The device to which the computer delivers information.

Patch (noun). A section of coding inserted into a routine (usually by explicitly transferring control from the routine to the patch and back again) to correct a mistake or alter the routine.

Positional Notation. One of the schemes for representing numbers, characterized by the arrangement in sequence of digits which are to be interpreted as coefficients of successive powers of an integer called the base of the number system.

In the *binary* number system the successive digits are interpreted as coefficients of the successive powers of the base 2, just as in the *decimal* number system they relate to successive powers of the base 10.

In the ordinary number systems the digits are symbols which stand for zero and for the positive integers smaller than the base.

Names of number systems with base from 2 to 20: binary, ternary, quaternary, quinary, senary, septenary, octonary (also octal), nonary, decimal undecimal, duodecimal, terdenary, quaterdenary, quindenary, sexadecimal, (also hexadecimal), septendecimal, octodenary, novendenary, and vicenary.

* Institute of Radio Engineers definition.

The sexagenary number system has a base of 60. The commonly used alternative of saying "base-3", "base-4", etc., in place of ternary, quaternary, etc., has the advantage of uniformity and clarity.*

Random Access. Access to storage under conditions in which the next position from which information is obtained, or to which it is delivered, is in no way dependent on the previous one.

RCA Character. See the RCA 501 Code, Appendix II.

Rerun. Rollback. See Appendix I.

Rewind. Move a tape in a backward direction.

Rollback. See Appendix I.

Sector. An area in the High-Speed Memory whose beginning and ending addresses are designated by the instruction.

Sign Position. The location to the right of the least significant digit of an item.

Standard Memory Locations. Designated locations in the HSM which are used for Address Modifiers, automatic storage of the final contents of the A Register, etc. (See Appendix VII).

Start Message Symbol. See Organization of Data on Tape (text).

Start Time. Time between the command to start an input-output device and the reading or writing of the first character.

Storage (Memory). A device into which units of information can be transferred, which will hold this information, and from which the information can be obtained at a later time.

Internal Primary Storage. Storage facilities forming an integral physical part of a computer.

Location. A storage position in the High-Speed Memory. Each location has a specific address and can hold one RCA character.

Register. A storage device with a specifically assigned function and a given unit capacity. Registers in the Computer in the RCA 501 System are of one, three or four-character capacity.

External (Secondary) Storage. Storage facilities which are not an integral part of the computer proper, but comprise units of the data-processing system (magnetic tape, paper tape, etc.)

Working Storage. A portion of the internal storage reserved for intermediate and partial results during computation.

Symbols, RCA Code. See Appendix II.

Tetrad. A unit consisting of four consecutive HSM locations or the contents thereof. A tetrad starts in a $(-----0)_8$ or $(-----4)_8$ address and ends in $(-----3)_8$ or $(-----7)_8$ address, respectively. A tetrad address, however, is any one of its four location addresses.

Unwind. Move a tape in the forward direction.

Variable Item Length. See page 8.

Word (in Electronic Computers). An ordered set of *characters* comprising the normal unit in which information may be stored, transmitted, or operated upon in a fixed-word or fixed-variable-word computer.

---

* Institute of Radio Engineers definition.

APPENDIX IX. ABBREVIATIONS USED IN TEXT

ABE	A Counter and B Counter Equality Flip-Flop
AOR	Adder Output Register
BA	Binary Add (41)
BRF	Block Read Forward (15)
BRR	Block Read Reverse (05)
BS	Binary Subtract (42)
BTC	Beginning of Tape Control
CIG	Character present in the Gap
CSG	Control Simultaneous Gate (75)
CTC	Conditional Transfer of Control (61)
DA	Decimal Add (51)
DD	Decimal Divide (54)
DM	Decimal Multiply (53)
DS	Decimal Subtract (52)
ED	End Data Symbol
EF	End File Symbol
EM	End Message Symbol
EMP	Electro-Mechanical Printer
ETW	End of Tape Warning
f	Used as subscript to denote "final"
HSM	High Speed Memory
i	Used as subscript to denote "initial"
ISS	Item Separator Symbol
IT	Item Transfer (21)
JR	Justify Right (33)
KC	Thousand Characters Per Second
L to R	Left to Right
LA	Logical "and" (47)
LNS	Locate *n*th Symbol in Sector (31)
LO	Logical "or" (46)
LRF	Linear Read Forward (14)
LRR	Linear Read Reverse (04)
LS	Line Shift
LSC	Least Significant (or rightmost) Character
LSD	Least Significant Digit
LW	Linear Write (12)
MSC	Most Significant (or leftmost) Character
MSD	Most Significant Digit
MSW	Multiple Sector Write (13)
NO	Normal Operation (Register)
OCT	One-Character Transfer (22)
PA	Paper Advance (03)
PC	Page Change
PET	Physical End of Tape
PR	Print (02)
PRI's	Previous Result Indicators
PRN	Previous Result Negative
PRP	Previous Result Positive
PRZ	Previous Result Zero
PS	Potentially Simultaneous
R to L	Right to Left
RAI	Return After Interrupt (77)
RD	Random Distribute (27)
RNS	Rewind *n* Symbols (16)
RWD	Rewind to Beginning of Tape Control (17)
SC	Sector Compare (43)
SCC	Sector Clear by Character (34)
SCD	Sector Compress— Delete Redundant ISS's (37)
SCR	Sector Compress— Retain Redundant ISS's (35)
SCT	Sector Clear by Tetrad (36)
SET	Set Register (72)
SM	Start Message Symbol
SO	Simultaneous Operation (Register)
SR	Select Read (Register)
SSG	Sense Simultaneous Gate (65)
SSM	Sense Simultaneous Mode (62)
SSW	Single Sector Write (11)
ST	Stop (76)
STA	Store A Register (automatic storage of final contents of A Register)
STC	Sector Transfer by Character (24)
STP	Store P Register (automatic storage of contents of P Register)
STR	Store Register (73)
STT	Sector Transfer by Tetrad (26)
SW	Select Write (Register)
TA	Tally (66)
TC	Transfer Control (71)
TCA	Three-Character Add (44)
TCS	Three-Character Subtract (45)
TCT	Three-Character Transfer (25)
TS	Tape Sense (63)
UNS	Unwind *n* Symbols (06)
VT	Vertical Tabulation
ZS	Zero Suppress (32)

NOTES

Form P501-2

Printed in U.S.A.