Functional Description of the NCR 304
Data-Processing System for
Business Applications

M. SHIOWITZ \hspace{1em} A. A. CHERIN \hspace{1em} M. J. MENDELSON

The National Cash Register Company (NCR) 304 system is a moderate-sized electronic data-processor for general business use. It incorporates many new design concepts which result in performance capabilities comparable to those of a large scale system. Each of the features was designed to provide maximum versatility in the handling of business data. The NCR 304 system offers the advantages of electronic data-processing to many organizations that have not yet found it economically feasible.

System Components

The major components of the NCR 304 system are depicted in the system block diagram of Fig. 1. They include the following:

INPUT DEVICES

A variety of keyboard data-entry devices which capture input information on punched-paper tape as a by-product of essential decentralized transaction recordings; these include cash registers, unit media readers (for reading price tags, customer tokens, clerk tokens, etc.) accounting machines, adding machines, window posting machines, and electric typewriters; all these devices are equipped with punched-paper tape recorders, and programming plugboards which permit flexible data arrangement.

CONTROL CONSOLE

The control console provides the power control, indicator lights, and operating switches for all units in the system which are connected directly to the central processor. The operator's control panel contains a lighted display for processor operation and for the operation of auxiliary equipment, and a paper-tape or keyboard operated electric typewriter which provides direct communication with the processor memory.

CENTRAL PROCESSOR

The Central Processor controls and, to a large extent, executes the stored program of data-processing operations. The high-speed main memory is of magnetic-core construction with a storage capacity of 1,000, 2,000, or 4,000 ten-character words. Transistors and magnetic cores are employed throughout as logical elements because of their low power requirements, low heat generation, high reliability, and compact size.

Access to the main memory is fully parallel, one word at a time. Processing is carried out in parallel, in which a processor word of a word being treated in parallel, and the characters of a word treated serially. The basic repetition rate of the processor is 400,000 characters per second. A word is accessed, processed, and stored (or restored) in 60 microseconds, constituting a minor cycle of the machine.

PAPER TAPE READER

High-speed photoelectric punched-paper tape reading is provided by the NCR Model 360. For flexibility, the reader is capable of reading five, six, seven, or eight channel tape. Any punched-hole code can be translated by the reader provided it contains fewer than 65 distinct characters. The reader operates on an individual character stop-start basis with a continuous reading rate of 1,000 characters per second.

Punched Card Reader

The NCR Model 380 photoelectric card reader provides high-speed punched card input. The reader is capable of handling 80-column IBM (International Business Machines, Corp.) cards. Provision is made for reading a portion, or the full 80 columns, of each card. Each card column may contain numeric, alphabetic, or symbolic punching. Card reading is performed at the rate of 500 cards per minute.

MAGNETIC TAPE FILE

The NCR Model 330 magnetic-tape file provides large capacity auxiliary storage. Data is stored on the tape in the form of records which may be of variable length. Each 2,400 foot-reel of tape stores approximately 4,300,000 characters. Information is recorded on tape at a density of 150 characters per inch, without gaps, between records. The tape moves at 100 inches per second with an acceleration time of 5 milliseconds. The information transfer rate is 15,000 characters per second. A minimum file consists of one control unit with up to eight tape handlers under its control. As many as eight controller units can be employed in a single system.

LINE PRINTER

The NCR Model 340 line printer produces printed output either directly from the processor memory or from magnetic tape. The printer is capable of printing 120 characters per line at a rate of 600 lines per minute. Fifty-six different characters may be printed. All paper spacing functions where no printing is required are performed at the rate of 72 lines per second. Multiple copies may be produced efficiently through the use of NCR carbonless paper, or standard carbon interleaved forms.

MULTI-PURPOSE CONVERTER

The NCR Model 320 converter transcribes records from one form of storage to another. Specifically, the converter can perform the following transcriptions: magnetic tape to printer, magnetic tape to punched paper tape, magnetic tape to punched cards, and punched cards to magnetic tape.

PRINTER-CONVERTER

The NCR 322 converter transcribes data from magnetic tape to the high speed printer.

PAPER TAPE PUNCH

The NCR Model 370 paper-tape punch provides punched paper-tape output directly from the Model 304 central processor, or from the Model 320 converter. Two different holes codes may be obtained at a punching rate of 60 characters per second. One of these codes is the processor code and the other is optional, and may be specified by the user to conform with leased line communication equipment being used in integrated data-processing systems.
Card Punch

Punched card output is provided from magnetic tape through the converter to an IBM Type 523 Summary Punch. Card punching is performed at the rate of 100 cards per minute.

Central Processor Characteristics

Code and Number System

Six-bit alphanumeric characters comprise the basic code of the system. An excess-zero binary coded decimal number representation is used for numeric characters, with the fifth bit of the most significant decimal digit of a numeric field serving as an algebraic sign. For decimal arithmetic operations, operands, and results are regarded as absolute value and sign.

Addressing and Field Specification

Because of the variability of field size in business-data processing applications, both within a single application and among several applications, the 304 utilizes a rather unique language for addressing and specifying the variable size of units of information to be processed.

In addressing ten-character words in the 1,000, 2,000, or 4,000 word core memory, addresses consisting of three characters are used. To refer to addresses other than 000-999 (i.e., in machines with the larger core memories) use is made of an alphabetic character in the most significant position of the address.

A sequential group of from one to ten characters within a word comprise an addressable field. These partial word fields are specified by two decimal digits used in conjunction with the three-character word address; one being used to locate the least significant, the other the most significant, characters of the field within the word. In arithmetic operations each field is treated as an individual unit containing its own sign. Alignment of fields for proper operation is performed automatically by the machine.

Since a variety of multiple-word fields must be addressed and processed in business applications, an appropriate addressing scheme for each type has been incorporated in the command structure.

1. Lists of fixed-length multiple-word items; when a sequential group or list of items of common length are to be processed as a unit, the item length in words is specified within appropriate commands by a two or three character quantity. The total size of the list is determined in either of two functionally equivalent fashions: if the size of the list is a function of the data being processed, which is often the case in transfers of lists of items between the processor and the input transcribers or the magnetic file, size specification usually consists of a tally of the number of items in the list; if the list size is essentially independent of the data being processed and is generated by the programmer, it is generally indicated by the specification of a terminal word address which locates the end of the list. Accompanying the item length and an initial word address locating the first word of the first item, either the item tally or the terminal address completely specify the location, size, and structure of the list.

2. Variable-length multiple-word items: in those cases where the length is a function of the data, e.g., in multiple-word instructions and variable-length magnetic file records, the information specifying item length is contained in the item field itself.

Where the programmer generally specifies item length, it is contained in the instruction which calls for the operations on the item. Control words, such as a key in a sorting process, contained in multiple-word items are addressed by specifying their position relative to the first word of the item.

Relative Addressing

Every command can utilize a relative address facility to permit automatic selective modification of command addresses for the execution of repeated routines and other useful memory indexing operations, thus saving considerable command storage space and execution time.

Address Modification

With respect to all automatic processor operations, such as augmenting control number, index register modification, automatic tallying during command execution, etc., the word addresses are continuous and cyclic.

The circuitry associated with the addition involved in these operations is mechanized to count sequentially from the first thousand to the second, third, or fourth thousand words, (depending on memory size), and cycle back to 000 when the end of the memory is reached. Thus address modification is modulo the size of the memory.

Special Memory Cells

In addition to the main memory, 40 special addressable cells are provided. These special memory cells may be used as data addresses in any command.
but are not included in the cyclic portion of the memory. Six of these addressable cells have special functions and are used by the machine during normal operation. The functions of these special cells are as follows: one is used to store automatic tallies generated on input and magnetic file operations, and for results in certain arithmetic commands and is named the tally register; another is the control register which contains the address of the next command in sequence during automatic processor operation, as well as an unconditional jump address. Four additional cells are used to store information during an automatic monitoring procedure.

The remaining 34 cells are available for storage at the discretion of the programmer. Since the special cell addresses are not included in the normal modification cycle of the machine, information stored therein will remain unaltered unless referred to specifically and individually.

**Command Sequencing**

During the automatic processor operation commands are normally executed sequentially, the address of the first word of successive commands being obtained by augmenting the right-hand address part, or C address, of the control register by the word length of the command being executed. When a transfer of control or jump is to occur, the contents of the C portion of the control register are replaced by the jump address.

In addition to the conditional jumps or branches which occur as the result of explicitly programmed examination of operands, a number of automatic branch operations are possible when exceptional conditions arise in the data or status of the components handling the data media. The sensing of these conditions is essentially a motor function performed by the logical elements of the control section of the processor. This automatic branch facility provides for execution of a number of control decisions inherent in all applications without explicit programming of the test.

A self-linking feature of the processor provides each command with the optional ability to automatically cause an unconditional jump and storage of a link address. The unconditional jump address is taken from the middle address part, or B address, of the control register, and the link address is stored by an interchange of the B and C portions prior to the access of the next command. This control of command sequencing is specified by one of the bits in each instruction.

It is valuable in many areas of programming, such as the design of closed subroutines.

**Command Structure**

The general characteristics of the 304 commands are difficult to classify in terms presently prevalent in the field. Enlarging the definition of the term address to include, where applicable, the variable size as well as the location of the operand or field, it can then be stated that the 304 has a multiple-address code with the number of addresses per command ranging from three to eleven, with optional automatic relative address modification, automonitor control, self-linking, and automatic branching.

A majority of the commands in the 304 repertoire consist of two words; the structure and component parts of a typical two-word command are shown in Fig. 2, and are explained as follows:

I and K, (each one character) specify the operation to be performed; the sign of K controls the self-linking characteristic described above.

A, B, C, (each three characters) are normally used as operand word addresses.

M, (one character) specifies the level of automatic program monitoring to be executed during processor operation in the monitor mode. Optional operator control of automatic monitoring is permitted by a switch on the control console.

R, (one character) specifies the index register to be used for relative modification of addresses A, B, C prior to execution of the command.

S, (one character) selectively specifies the combination of addresses A, B, C, to be modified by the corresponding contents of R.

AL, AR, BL, BR, CL, CR (each one character) identify the left-most and right-most digit positions of the partial word fields within A, B, and C respectively.

Since the use of the relative address and partial word modifiers are optional, the total effective operand will be briefly referred to hereinafter as A, B, or C.

**Internal Two-Word Commands**

The command list of the central processor contains 22 internal operations which are specified by two words of instruction information:

**Decimal Arithmetic Commands**

The five operations comprising this group are:

1. **Add:** A is added to B, the sum is stored in C.
2. **Subtract:** B is subtracted from A, the difference is stored in C.
3. **Multiply:** A is multiplied by B, the most significant portion is optionally rounded and stored in C, and the remaining portion of the product is stored in the tally register.
4. **Divide integers:** B is divided by A, the integer quotient is optionally rounded and stored in C, and the integer remainder retaining the sign of B is stored in the tally register.
5. **Divide fractions:** B is divided by A, the rounded quotient is stored in C.

The signed numeric operands have their decimal point immediately to the right of the least significant digit in operations 1, 2, 3, and 4. In the divide fractions command, the decimal point is immediately to the left of the most significant digit. In commands 3 and 4, the rounding option is specified by instruction character K; the digit position in which rounding occurs is automatically determined by the number of digits allocated to result storage in C.

Overflow occurs in those cases of execution of add, subtract or divide integers commands where the number of significant (non-zero) digits of the result exceeds the capacity of the storage field specified by the digits, CL, CR.

In this event, an overflow alarm indication is automatically stored in the control of the machine, and unless the next command executed is a branch command which tests this overflow alarm, an error halt occurs.

**Non-Decimal Arithmetic Commands**

Because the arithmetic requirements of address modification and other program control operations in multi-address computers differ markedly from those of decimal calculations involving business data, the command code includes three non-decimal arithmetic instructions.

Two operations expressly designed to permit flexible and efficient memory indexing are:

6. **Modify add:** A is added to B and the sum is stored in C.
7. **Modify subtract:** A is subtracted from B and the result is stored in C.

These operations are modular with respect to memory size, and all carries are suppressed between adjacent triads (3-character addresses) in multi-address operand fields. Hence the execution of a single additive or subtractive modification can provide for the simultaneous indexing of up to three addresses.

The usefulness of the relative address facility is further enhanced by the following command:
Assembled and the resultant combination cells beginning at codes is not offset by the requirement Shiowitz, Cherin, Mendelson—NCR 304 Data-Processing System

COMPARE COMMANDS

The three compare operations include:

9. Compare numeric: A and B are compared algebraically, and a branch to C occurs if A is greater than B.
10. Compare alphanumerically: A and B are compared alphanumerically, and a branch to C occurs if A is greater than B.
11. Compare equality: A and B are compared alphanumerically, and a branch to C occurs if A is equal to B.

Due to the nature of sign representation in the processor, execution of command 11 with numeric operands will result in a branch if and only if the signs as well as the absolute values are equal.

EDITING COMMANDS

Five commands were designed for fast flexible editing and rearrangement of data, which are necessary procedures in all business programs.
12. Compress: the fields A and B are assembled and the resultant combination field is stored in C.
13. Distribute: the field A is distributed into (the partial word field storage locations) B and C. An optional sign split-off in the field distributed is specified by the value of K. Sign split-off consists of generating a nonsignificant or signed-zero character, and inserting it in the extreme left character position of the stored field. (Sign compression, which is the inverse operation, occurs automatically as a result of logical mechanization.)
14. Zero suppress: the field A is zero suppressed, and the result with optional sign split-off is distributed into B and C.
15. Edit: the field A is either zero suppressed, or check protected and distributed into B and C. The option of suppression or protection is exercised by specifying the value of K.
16. Transfer: the contents of a consecutively numbered block of memory word cells beginning at A, are transferred to a block of cells beginning at C. The number of words transferred may range from 1 to 1,000, 2,000, or 4,000, depending on memory size.

BINARY COMMANDS

In many programming areas the advantages in storage efficiency of binary codes is not offset by the requirement for frequent conversions to decimal or alphabetic representation. The processor design allows for optimum economy of hardware application in these areas by providing five commands of binary nature:

17. Extract: the bits of A are logically multiplied by the bits of B and the logical product is stored in C.
18. Insert: the logical product of A and B, and the logical product of C and the complement of B are formed. Then the logical sum of these two products is formed and stored in C.
19. Test Bit: a bit by bit test of A and B is made; if A has a bit value of one in every bit position in which B contains a bit value of one, a branch to C occurs.
20. Pack: this command packs a group of consecutive three-word blocks of positive-numeric data into a group of consecutive two-word blocks by dropping the two most significant bits of each six-bit character. It should be noted that for positive-numeric data, these bits are uniformly zero.

TEST COMMAND

The final command in this list is a general-purpose jump command:

22. Test: this command tests the status of a set of control-console switches, the paper-tape punch, the paper-tape reader, or the overflow alarm, as determined by the value of K. The program can branch as a result of the test, thereby permitting operator intervention to control program execution.

INTERNAL MULTIPLE WORD COMMANDS

The NCR 304 includes a set of instructions which are equivalent to complete subroutines in machines with conventional codes. These commands have been carefully chosen to be of general applicability to all business data processing, and have been designed to insure maximum flexibility. The usage of a single command to carry out a substantial processing subroutine provides considerable savings in execution time and storage requirements over conventional programmed subroutines. Three of these are internal multiple-word commands.

23. Merge: this command merges two distinct sorted groups of consecutively stored, multiple-word items (groups A and B) into a single sorted sequence (group C). The data in groups A and B must be standardized such that each item contains the same number of words, and the control key characters (the numbers which determine the sort sequence) must occupy the same relative positions within each item. It is however not necessary to have the same number of items in groups A and B.

This command may be executed in two alternative fashions: one mode is used when the data to be sorted can be completely stored in the internal memory; the other mode is used when the volume of data exceeds the memory capacity and magnetic tape is used for auxiliary storage. In either mode, the command is specified by six words of control information, in which the programmer specifies the following variables:

The location of the first words of the first item of group A and group B.
The size of each group (A and B).
The number of words per item.
The location of the first word of the first item to be put away in group C.
The size of group C.
The mode of operation (internal or external).
The number of words containing the sort-key (one or two words).
The relative location within the item of the word (or words), containing the control key, and the location within these words of the characters comprising the control key.

A set of three alternate addresses, one of which may be selected to specify the location of the next instruction depending upon the status of the data at command termination.

When the command is executed in the internal mode, both group A and B are fully merged and stored as group C before the command is terminated. No branch occurs in this mode. When the command is executed in the external mode, the command will terminate when any of the following occur: group A is exhausted, group B is exhausted, or group C is filled. A different branch address is chosen in each case, which permits the introduction of new data from magnetic tape in the first two alternate manners or the storage of group C on magnetic tape in the third alternative.

After the appropriate tape operation has been carried out, subsequent execution of the same merge instruction will automatically pick up the process at the same point (in the non-exhausted groups) where it was interrupted.

24. Sift: this command assumes the existence of a sorted group of multiple-word items stored in consecutive memory cells. The command compares specified control characters of each successive item with those of a “standard” item. The number of items whose control key is less than or equal to the control key of the standard are tallied and stored. When the first item is found whose control key is greater than that of the standard, or when a specified terminal address is reached, the sift command terminates and the processor proceeds to the next normal command. This instruc—
tion is specified by three words of control information containing:
The location of the first word of the first item to be sifted.
The location of the first word of the standard item: this standard may itself be a member of the group to be sifted.
The number of words in each item.
The location of the control words within each item and the location of the control characters within each control word.
The location of the put away for the sift tally.
The total number of items to be sifted.

This command may be used in two distinct applications. As a table look-up command, the tally provides the location of a desired table entry. During report preparations, this command, in conjunction with the summarize command (see operation 25) permits the rapid accumulation of control totals.

25. Summarize: this command calculates algebraically the contents of specified fields contained in successive multiple-word items stored in memory. This command is specified by three words of control information.

The location of the summarization field in the first item.
The number of words per item.
The storage location of the accumulated total.
The location of the number of items to be summarized.

The tally which is generated by the sift command (as explained in operation 24) may be directly referenced in that portion of the summarize instruction which determines the number of items to be summarized. These two commands, used in conjunction, therefore automatically provide a summarization of all those items whose control keys are identical, which is precisely the objective of summary report preparation.

Input-Output Commands

Nine two-word commands enable the NCR 304 Processor to communicate with and control its associated input-output equipment; these commands provide the ability to read and write on magnetic tape, search magnetic tape, read and punch paper tape, read punch cards, and print on the high speed printer.

Reliability Checking

The error checking that has been incorporated into the 304 system is based on the philosophy that checking should be performed whenever errors are probable, and that the thoroughness of the check should be a function of the probability of the occurrence and the relative importance of the error.

The types of checks that exist in the system can be classified into read-write checks, transmission checks, and functional checks. The read-write checks are performed on all input-output information that enters or leaves the system. Transmission checks are performed whenever information is transmitted between components within the system. Functional checks are performed by the individual components to insure the correct execution of their functions.

The errors that are detected by the 304 system are roughly divided into two classes. The first class contains errors that might be corrected by programming. For example, read errors on magnetic tape, where errors are assumed to be caused by foreign particles on the tape, might be corrected by re-reading.

Detection of these errors does not necessarily halt the operation, but allows it to continue until some reference point is reached. Where appropriate, an automatic program branch occurs, and the programmer is given the option in an attempt to correct the error of repeating the operation, of halting the equipment and signalling the operator, or of noting the error and continuing.

The second class of errors is characterized by the fact that they cannot be corrected by programming. An example would be the detection of a broken tape on a tape handler. This type of error requires operator intervention; the operation is accordingly halted in these cases, and no programming option is provided.

Summary

Effective application of a business data-processing system dictates that the system design provide maximum flexibility in programming the solution of a problem. On the other hand, economy of programming effort and of hardware utilization can best be achieved by rigidly structuring the automation of the data handling involved in, and the control of, those processes which are common components of broad classes of problem solutions.

Conventional system designs usually meet only one of the two foregoing requirements. If a system meets the former, it is referred to as a "general purpose" system, and, if the latter, a "special purpose" system.

Programmers have indicated recognition of these conflicting requirements, and have attacked the portion of the problem concerned with conservation of programming effort by developing a number of techniques. For example, many programmers are presently using various pseudocoding techniques, in conjunction with generator programs, in an attempt to provide a workable scheme for the reduction of general programs to detailed machine codes for conventional general purpose systems. Application of this resultant combination of hardware and pseudo codes achieves a reduction in programming effort, but no economy in hardware utilization is obtained, since the computer time spent in processing control information in the resultant routines is essentially unchanged, and data transfers are often needlessly repeated.

To resolve the conflicting requirements outlined above, the design of the 304 system is based on simultaneously achieving the advantages of both general and special purpose systems. The automation of processes, and of the control of processes, which are common to all business data-processing problems (such as sorting, summarizing, editing, etc.), is a characteristic of the 304 system which gains the advantages of the automaticity inherent in special purpose designs, without any loss in programming flexibility.

For a given process and information transfer rate, execution time is reduced because the specific transfers of, and operations on, the data and control information which occur are tailored to the specific needs of the process. Complete programming flexibility is retained not only within the structure of the electronic subroutine commands themselves, but also by the inclusion of a full complement of commands of a general purpose nature.

Discussion

C. B. Poland (General Electric Company): On off-line copy, what happens when a new record must be inserted between two old records?

Mr. Cherin: The copy operation as executed on the 304 is one of proceeding from the present position, so to speak, in the file to a new position and stopping at that point, that is, copying until a specified control configuration has been reached. In all processing, some identification is required for each record and this identification is used to control the copy. Thus, specifically, because the programmer knows that he must insert a new record, and as a result of this he knows what this record's control configuration, address, or account number for...
A Technique for Using Memory Cores as Logical Elements

L. J. ANDREWS

OVER THE YEARS there have been many outstanding papers which can be assembled under the collective title "Component Failure Analysis." Included in this group are the tube-life prediction tables, the derating charts for condensers, the maximum current versus useful life curves for diodes, etc.; and each company that has a customer service organization has in its files records loosely titled "Plug-in Failures, Their Cause and Cure." An examination of the records will show that in the majority of cases the "active" elements are at fault. Active elements are defined here as those elements which amplify a changed state of their inputs. It would seem, then, that the path to reliability is to remove as many active elements as possible from the system; that is, given some specific design problem, to time-share the active elements as much as possible in keeping with the flexibility required of the overall design. But this philosophy is not without its attendant apparent disadvantage. To use a minimum of active elements, a maximum of switching elements is required. To utilize the minimum active elements concept, a switching element approaching the ideal is required.

The ideal switching element should have the following characteristics: 1. there should be a minimum number of passive elements per switch, 2. the switch should not load the logical propositions, 3. the switch should be lossless, 4. the switch should have high discrimination, 5. the switch should be compatible with other system components, 6. the switch should be easily fabricated, and 7. it should have all the miscellaneous properties such as high speed, high output, and small physical size; and should be insensitive, shock resistant, temperature insensitive, etc.

An available item that to a reasonable degree fulfills the miscellaneous properties is the small ferrite core usually used in memory applications. How these cores can approach the ideal switch and perform other useful functions as a result of a unique system concept is the subject of this paper.

The Inhibit-Wound Core

The inhibit-wound core is a deceptively simple component. In its rudimentary form it consists of a small ferrite core with one or more driving sources, an inhibiting proposition, and a sense winding. Each "winding" consists of a single wire through the core; the clocks and propositions each carry half-select current. Using the mirror symbols for core-winding senses it is apparent that the core in Fig. 1 will change state from a one to a zero and back again as the \( C_a \) terminal alternates between positive and negative potential. A one is defined here as the up direction when current flowing into a slant bar is seemingly reflected up. A zero is thus also defined as current into a slant bar reflected down the core. The polarity of the sense signal need not be specified at this time.

Mr. Cherin: I don't believe I made the point quite clear. The computer, including the memory, is mechanized completely from transistors, diodes, and cores; the cores in particular are driven with transistors.

Mr. Bindloss (Liberty Mutual Insurance): Does the 304 system have a file interrogation unit for determining the status of a particular item in the file on an unscheduled demand basis? If so, what is the searching and read-out time? How many tape drivers can the NCR 304 accommodate?

Mr. Cherin: Not as yet. The searching rate in such an operation would be, though, the same as the read or write rate which is 15,000 characters per second. The magnetic tape, as we all know, is not a very good medium to use for random-access interrogations. The system accommodates a maximum of 64 tape drivers per system.

R. Douthitt (Sperry-Rand Corporation): Does the sort command still exist in the 304?

Mr. Cherin: It has been renamed and is the merge command which contains the six-words of control information.

L. C. Hobbs (Sperry-Rand Corporation): I have previously heard the card rate of your machine described as 1,000 cards per minute, but you stated 500 cards per minute. Is this an interim unit?

Mr. Cherin: No. I would say that this is a conservative figure.

In Fig. 1, then, the clock \( C_a \) has the ability to write a one when positive, and can read a one or zero when negative. The previous one or zero state is detected during read by the presence or absence of a large flux change and corresponding voltage induced in the sense winding. The propositions \( X_i \) have the restrictions that they may have current only during the write phases of the clock, their currents may only be of half-select or zero amplitude, and their currents will be only positive, i.e., away from the terminus of Fig. 1, and then only if the proposition is true. The relations between the propositions and the clock phases is best illustrated by referring to Fig. 2. During time \( T_b \), proposition \( X_i \) is true, current is then flowing in the wire \( X_1 \) in the direction previously designated as positive. The proposition \( X_i' \) (if it exists for use elsewhere) does not have current during the \( T_w \) write time. Conversely, during \( T_1 \) and \( T_2 \) the wire designated \( X_i \) carries no current while wire \( X_i' \) has half-select current during write periods. It is a natural consequence, of course, that the core of Fig 1, wired with proposition \( X_i \) and subjected to the wave forms of Fig. 2, will have the states 0, 1, 1, and 0 at the end of timing periods \( T_b \), \( T_1 \), \( T_2 \), and \( T_3 \) respectively. The core then can be said to have taken the complement of the information held in sequence by the proposition \( X_i \) within the framework of the system defined. Of more immediate interest is the case where not \( X_i \) but \( X_i' \) is the proposition present as an inhibitory signal. Here the core will assume information as repre-

L. J. ANDREWS is with the Electronics Division of the National Cash Register Company, Hawthorne, Calif.

The author expresses his appreciation to Walt Edwards and James Hudson, who supplied the initial impetus for this work.

Andrews—Using Memory Cores as Logical Elements

From the collection of the Computer History Museum (www.computerhistory.org)