M. L. Lesser
J. W. Haanstra

The Random-Access Memory Accounting Machine

I. System Organization of the IBM 305

Introduction

The logical organization of the new IBM 305 Random-Access Memory Accounting machine (RAMAC®) permits the processing of business transactions as they occur. The engineering design features of the random-access system are shown in this paper, and the advantages over sequential-access systems in automatic accounting machines are shown. This new system utilizes the IBM 350 Magnetic Disk Random Access File, which is described in Part II of the paper.

The first section of Part I discusses the systems study of this "in-line" concept in business applications which led to the design of the IBM 305. In particular, it was found that arithmetic power, per se, was very much secondary to the ability to manipulate variable field lengths of alphanumeric data in a direct fashion and to make direct multi-choice decisions based on the current status of the information being transferred. It was also considered necessary that the programming technique be as straightforward as possible in order to eliminate the necessity for a high degree of special training on the part of potential users.

The second section of this paper discusses the actual logical organization of the machine. The data transfer paths are shown, along with the decimal arithmetic system. These include the communication links between input, processing, magnetic disk file, and output. The control system is discussed, with emphasis on the ability of the machine to overlap operations occurring in these several components.

The "in-line" processing concept

Historically, the processing of business data originated as an "in-line" operation. We might say that with the earliest direct barter systems, data processing began and ended when the bartered material changed hands. Thus, the operation was not only "in-line," but was also "immediate."

As more formalized business-record systems come into

Abstract: The design features of a new automatic data processing machine for business applications, utilizing a random-access memory system, are described. Unlike the usual "batch" method of machine-processing business transactions, the technique used permits transfer of information between any two points in the system and allows multi-choice decisions according to the current status of the information. The "in-line" operational concept is discussed in detail and the data transfer routes and processing controls are shown. Employing punched-card input and printed-record output, the IBM 305 accounting machine is designed to handle 10,000 line-transactions per day.

being, data processing lost some of its immediacy but still remained more or less an in-line affair. In general, transactions were posted to their final record forms in the sequence that they occurred, preferably with no intermediate posting operations, and most certainly without awaiting the accumulation of sufficient similar transactions in order to make an economical run of data. In modern machine accounting terminology, there was no sorting or collating except by distribution directly to the appropriate ledgers.

With the steady increase in volume, the manual in-line process could not keep up with the demands for relatively current data in an easily digestible form. The solution to this problem has been the mechanization of business data processing by various devices and in various degrees. Inherent in most modern mechanization schemes of any magnitude, to date, has been the "batch" concept. According to the traditions of mass-production technology, the process of mechanization was approached by the route of many simple repetitive operations. An array of single-purpose machines was built and the information passed through them in turn. A little more of the job was bit off with each machine pass, until the final report could be assembled from all the individual sub-assemblies which had been fashioned on the machines. The time necessary to accumulate a batch was accepted

as a small loss in view of the large time savings available by the mechanization of data processing. Some continue to accept this as a "way of life," even in the face of evidence to the effect that the batch accumulation time is now growing longer than the process time in some applications not inherently adaptable to batching. Faster and more costly devices are now being built in an attempt to salvage a sense of immediacy from data that grew cold even before they were presented to the machine. This is done by building multi-purpose electronic machines that carry out all the operations formerly handled by the battery of single-purpose machines. Although all this is done within a single set of cabinets and with no manual intervention between operations, nothing has really been done to eliminate the necessity for batching.

A typical flow chart for a job handled with conventional punched-card equipment is shown in Fig. 1. The job is one of billing and invoicing, with the maintenance of a perpetual inventory, and is the source operation for data going to "accounts receivable" and "sales statistics analysis" operations. The number of files to be consulted is small. Only the inventory file is maintained by the system in any way other than to replace the information removed from it. The "sorts," "collates," and "card moving" stages are all shown. If the flow chart is translated to tape terms, the remaining sorts and collates are still necessary. Most of the "decisions" required are not shown and will be discussed later.

However, if the restriction that file information is available only in a fixed, predetermined serial order is removed, the batching requirement is eliminated. If there is *random access,* with equal facility, to any place in the memory, it is possible to pick out directly the necessary items from each of the reference files in the sequence established by the order in which the input data are received and the secondary order established by the transaction. In theory, a machine can be built to take an input transaction record and carry it all the way to final output document, distributing by-product information to the proper files en route, with no regard as to whether or not the next input refers to the same type of transaction. The flow chart would then look like Fig. 2. With this ability, "batch" processing techniques can be reserved for those applications, such as statistical analysis, that have an inherent batching requirement.

The memory device required to handle a large-capacity in-line processing system is available. The IBM Type 350 magnetic-disk, random-access memory, is described in Part II of this paper. This memory is a device allowing live storage with access to any one of 50,000 100-character alphanumerical records in approximately one-half of a second. It should be noted that the division into 100-character records is for convenience only; if the application requires it, any desirable fraction or multiple of this record length may be used and intermixed in the file, with a corresponding adjustment of the number of records available.
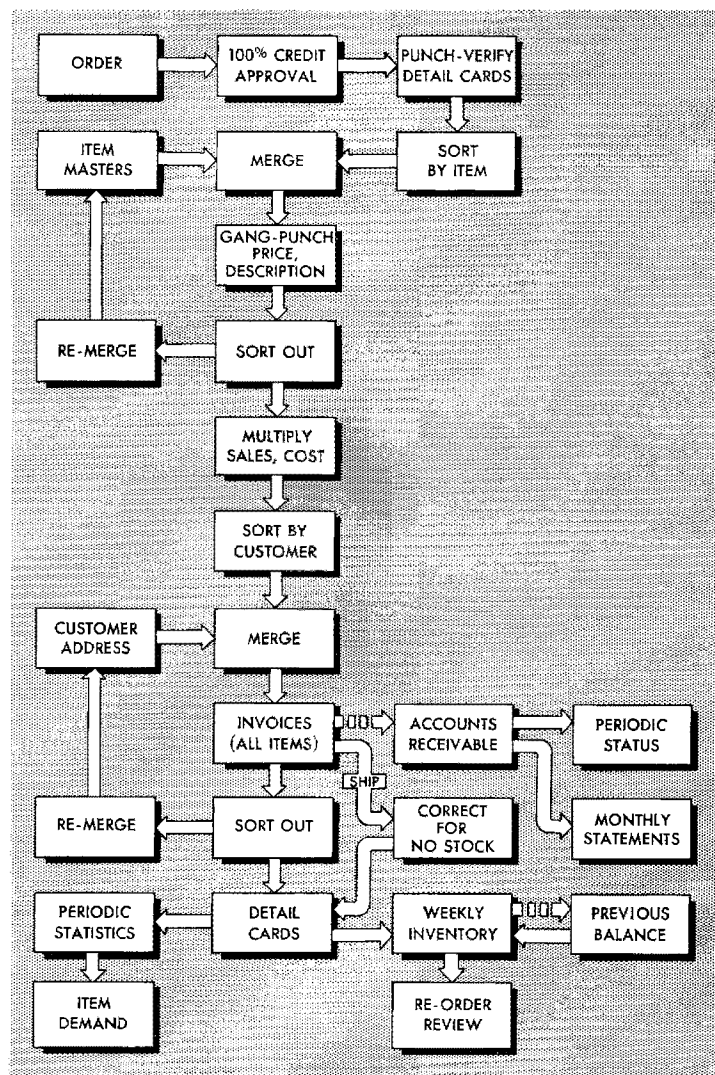

Figure 1
Typical IBM batch processing.

Machine system requirements for in-line processing

In this part of this paper we will discuss the system requirements established for the complete data processing machine, IBM Type 305, designed specifically to make use of the disk file. The detailed characteristics of the machine will then be described. Only the gross aspects of the system will be considered at first, as indicated in the block diagram shown in Fig. 3, with emphasis on the desired characteristics of the unit marked "process" in that diagram. Physically, this is a small magnetic drum carrying multiple tracks, each capable of storing a single 100-character record, and surrounded by a control system. We will neglect the input-output system, and the file itself, except for pointing out that they are connected to the process unit by information channels and are so buffered to the process unit that all
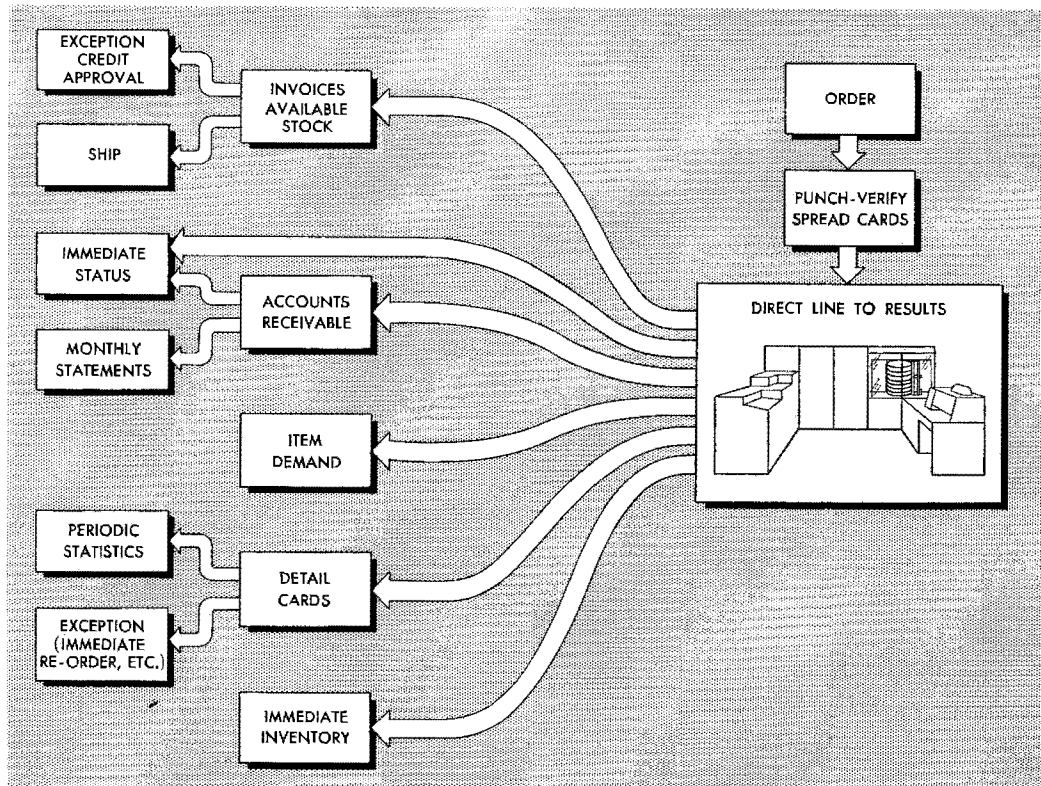
63

Figure 2
Typical IBM in-line processing.

mechanical time delays may be overlapped by process operations.

Briefly, the original requirements for the machine as designed were as follows:

1. An in-line accounting machine that would handle, from card input to printed-record output, 10,-000 line-transactions per day. This implies posting the basic data for each transaction to the record or records involved, thereby maintaining all pertinent files on a current basis as a direct adjunct to transaction processing, and also making all those routine decisions that could be relegated to an automatic machine. The intention is to approach "management by exception," where only those decisions essentially requiring manual handling are brought to the attention of the operators.

2. During the processing of these transactions, the system was to allow manual inquiry as to the status of any record in the disk file at any time, producing a printed output independent of the primary output from the machine.

3. The whole machine system was to be made available at a cost that would be considered reasonable to a user for whom 10,000 transactions was a very large day's business.

A secondary requirement that was observed throughout the system design was that the operating procedure of the machine should be as straightforward as possible. The form of instruction chosen should be as suited to

the task as possible; there should be a minimum of programming restrictions chargeable to the machine, as opposed to those chargeable to the application; and the operating procedure, in general, should be easily understood by persons already familiar with conventional punched-card techniques.

In the design of a machine system, practically the entire arrangement centers around the method chosen for control. In general, previous flexible machine systems were based on one of two techniques — a fully stored program, or a fully wired, control panel program. For the IBM 305, our approach to the control problem was based on a study of the operation the machine would be required to perform. When a conclusion was reached as to the form desired (an optimum combination of the two techniques), the general organization requirements of the machine followed directly.

In an accounting operation sequence, such as shown in Fig. 1, the majority of operations are of the reproduction type, consisting of straight information transfers, with a relatively small amount of computation. There are also a large number of multi-choice decisions to be made, most of which are not shown explicitly in that illustration. In any case, it appeared that the proposed control system should be slanted toward the ability to mass-transfer large quantities of variable field-length information, and the arithmetic requirements should be met as an adjunct to this ability.
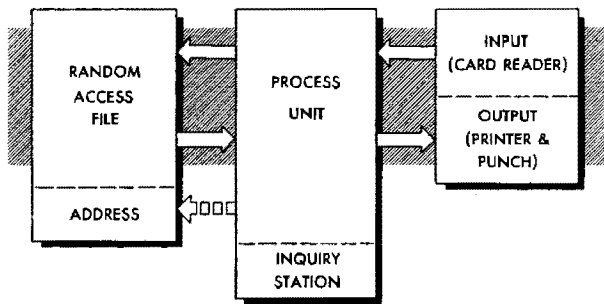
Figure 3
Gross block diagram of the IBM Type 305.

In addition to the transfer and arithmetic operations, a machine system requires the ability to make three distinct types of decisions if it is to operate successfully as an automaton. These are:

1. *Decisions based on static information wholly contained in a given record.* These might be considered as "type of record" decisions and can usually be handled most efficiently by recognition of a code mark or character in the record. Thus, for example, the machine can make a decision based on the type of input card as to whether this transaction is an order, a stock receipt, or a customer payment record, and can go immediately to the program and procedure necessary to process that transaction.

2. *Decisions based on the status of the transaction processing to date.* These are usually the result of arithmetic operations and serve to indicate such basic items as "out of stock, check for substitute or back order;" "below minimum re-order level, signal for stock re-order;" "customer's credit allowance has been exceeded, so signal for manual credit authorization before shipment;" etc.

3. *Decisions based solely on the sequence of processing to date.* An easily understood example of this is the case of substitutions allowed for out-of-stock orders. If it is assumed that item *A* may be substituted for item *B* if necessary, and vice versa, the procedure must prevent a second substitution if, after the first is tried, the second item is found to be also out of stock.

It is recognized that machine systems have been built that handled all three types of decisions by means of a single decision element actuated as a result of arithmetic operations. However, for all decision types other than the second, this method increases the requirement for arithmetic operations not associated directly with the requirements of the problem, which, in turn, increases the complexity of programming otherwise straightforward applications.

It was early recognized that the most direct method of machine-actuated decision-making was through the use of the control-panel concept and direct relay-switching of control impulses. The concept of the "selector" is well understood, and separate selector systems could be provided for each of the three types of decision elements required. However, control-panel operation of the infor-

mation-transfer function becomes unwieldy for large amounts of relatively poorly-sequenced data, particularly if the nature of the problem eliminates the possibility of considering the information as being grouped in elements of fixed word length. Thus, to handle variable-length information fields purely on a control-panel basis, it would appear necessary to utilize the technique of the basic tabulating accounting machine — that of carrying an individual wire for each character to be transferred and utilizing mass selection to program the problem. While this offers several advantages for the limited amount of format change required in output printing, and is used in the system for that application, it does not appear feasible for the major information transfer system.

Stored-program concepts used to date also offered several disadvantages. Various solutions to the variable-field-length problem are available, usually based on a record mark either inherent in the record itself or programmed into the machine by way of an intermediate transfer register. Both of these approaches put artificial restrictions on the programming and, in general, introduce "program steps" that are necessary because of machine restrictions and do not contribute directly to the processing of the transaction. Decision-making via traditional stored-program concepts was too unwieldy even to be considered — as it usually is based on single binary choices resulting from real or induced arithmetic inequalities.

The basic control system for the IBM Type 305 is a compromise offering the best features of both the control panel and the stored-program control systems. All information transfers during transaction processing are made via a stored program. The necessity for record marks is eliminated by specifying the number of characters to be transferred as part of the instruction. Each ten-character instruction thus completely specifies a single reproducing operation. The track and starting column addresses of both the sending and receiving process drum tracks involved are written, followed by the two-digit quantity of the number of characters to be transmitted. Arithmetic operations, in the form of addition, subtraction, and distribution and multiplication (absolute values only) were treated as information transfers and programmed with the same form of instruction. Arithmetic entered into the basic format specification only because of our language inconsistencies. We read alphabetic information from left to right and do arithmetic from right to left. In order to propagate carries during arithmetic processes, the internal information transfers are made in a right-to-left manner. Thus, the starting column address for any transfer is the "low order" column of the field. The system is completely analogous to setting up card processing routines, with added conveniences, such as the ability to distribute a single quantity into up to ten adjacent accumulators on two instructions, one of which would probably be necessary for a later multiplication in any case.
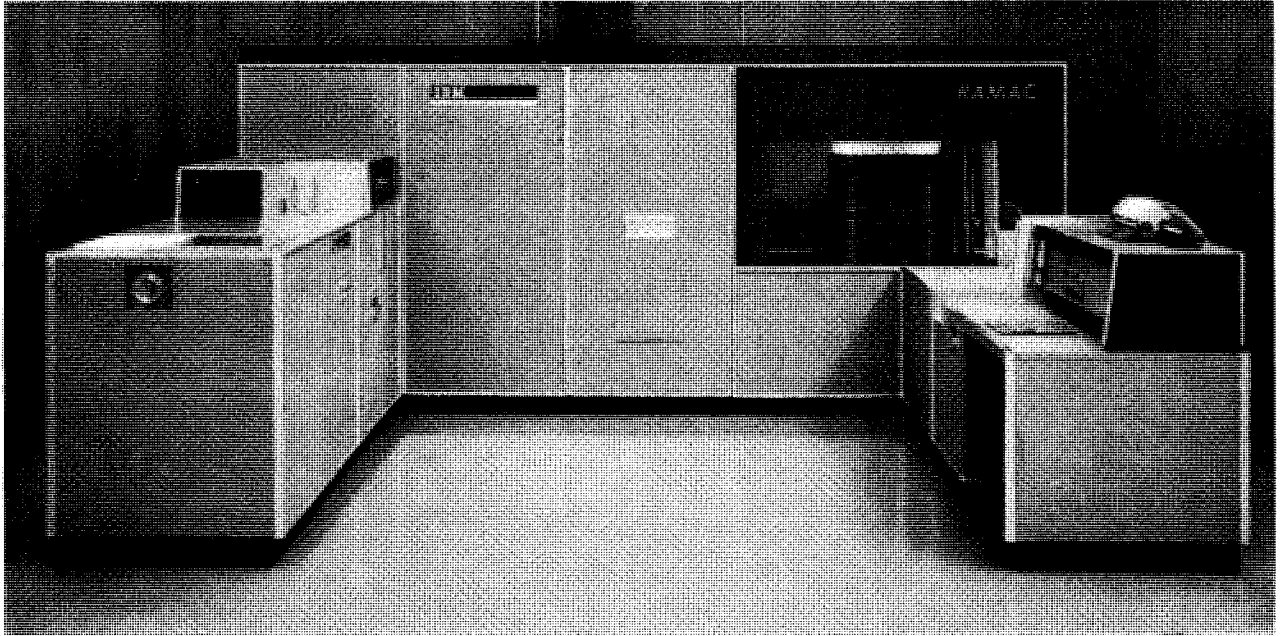
65

Figure 4
The IBM 305 Random-Access Memory Accounting Machine.

All logical decisions are made by means of selector wiring on the control panel. Process control may be passed at will between the two forms of control. Instructions follow in numerical sequence until after one is executed that is "flagged" by one of 47 symbols denoting transfer to the control panel. At this time, a control pulse is available from the corresponding one of 47 exit hubs on the panel. This pulse may be sent through a selector network and control returned to any instruction in the stored program, thus restarting the sequence. In this manner, multi-choice decisions are made on a single operating cycle to the control panel.

The available decision elements, as noted above, are all selectors. The three types of decisions are handled by three differently actuated sets of selectors. "Recognition" decisions are made by means of a character selector that consists of points on a selector tree that is actually a single-character relay register. Any character in any record may be sent to this register, replacing its previous contents. This character then controls up to a 48-way choice by a single interrogation of the Character Selector.

"Status" decisions are made on several bases. Each of the ten accumulators carries its sign in relay storage, the points of which are available as selectors on the control panel. Thus, decisions may be made as to the status (plus, zero, or minus) of each of up to ten accumulators, at any time. Further status decisions may be made as a result of field comparison, identical or not, and on whether or not any significant information other than "zero" or "blank" was transmitted on the most recent information transfer. This last item is very convenient in spread-card applications.

"Positional" decisions may be made using latch-type

relay selectors that can be picked up or dropped out by control pulses on the panel. Means for pulse-delay are also provided to eliminate "relay races" in setting up selector sequences through the points of controlled selectors.

Although the machine has facilities for modifying its own stored instructions by internal operations, this is very rarely done. The main reason is that the instruction used is extremely powerful as compared to the conventional single-address operation, and all instructions are directly useful operations. Thus, the total number of instructions necessary to perform most transaction processing is usually relatively small. The secondary reason is that if the immediate instruction capacity of the process system should be exceeded for a given application, it is usually easier and faster to bring in ten new instructions as a single record than it is to perform arithmetic operations on one or two.

General system organization

Fig. 4 is a picture of the IBM Type 305 RAMAC. Physically, there are four distinct units within the system. At the right of the picture is the console which contains a card reader, switches and lights for control of the machine, and a keyboard and printer for file inquiry and system test. The unit in the center is called the main frame and consists of the magnetic disk random access file at the right, the processing unit behind the two center panels, and the power supply behind the panel to the far left. To the left are the output units. In the background is the Type 323 card punch and in the foreground is the Type 370 printer.

66

Fig. 5 shows the system organization of the Type 305 RAMAC. Information enters the system via the card reader shown in the upper right-hand edge of the diagram. The information is written directly from the reading brushes onto a magnetic drum housed in the processing unit. On the next card cycle the information is read at a check station and compared with the data on the drum. Following a successful check the information is available under program control within the process unit. With the use of two tracks for input it is possible to parallel card feeding and processing operations.

The processing unit contains a magnetic drum with 33 tracks of 100 characters each and the circuitry for manipulating information on the drum according to a program that is also stored on the drum. Twenty of the drum tracks can be directly addressed from the program counter and make 200 program steps directly available for use. Additional program steps may be brought from any storage in the system and the 20 program tracks may be used for general storage as well as program storage. In addition, there are four general storage tracks, three tracks used in the arithmetic system, a track for use with the keyboard and printer unit on the console, two tracks used for output, and a timing track. The Random Access File is connected to the processing unit at two points. The address register, which specifies the record to be read from the file, is a five-position storage unit. Numbers from any part of the process unit may be sent to this unit. The selected record of the file unit, as specified by the address register, is available to the process unit for reading to procure information from the file and for writing to place new information in the file. The results of processing are sent to the output track on the drum. The printer and punch take data directly from this track and print and punch them according to the rules set down on their respective control panels.

An inquiry system to allow calling records from the file independently of normal processing operations has been provided. The location of the required record is keyed into the keyboard, and at an appropriate time this record is procured from the file and placed on a special track into the process unit. Typing of the record from this track proceeds independently of normal processing.

Information transfer

Information transfer between the process drum and the file occurs in blocks of 100 characters. Information transfer within the process drum may occur in blocks of from 1 to 100 characters according to the stored program instructions. Fig. 6, which depicts the transfer of information between two tracks of the process drum and the input-output section, shows how the first eight characters of the basic instruction are used to specify this transfer of information. Fig. 5 shows the means depicted for executing the basic transfer instruction.

All information transfers are routed through a 100-column core buffer unit. For the execution of any given instruction, the input to the core buffer is switched to the track which is to deliver the information. On the reading cycle the path from that track to the core buffer is closed at a time corresponding to that when the first position to be read is under the head on that track. The path remains closed until the number of characters specified by the "length" in the instruction have been transmitted, at which time the path is opened. The output of the core buffer is switched to the track which is to receive information, and that path is closed and opened during the writing cycle according to the position where writing is to start and the length, as specified in the instruction.

The processing unit executes information transfer instructions in sequence, as specified by a program counter which controls the next instruction to be read. The instruction reading is indicated in the center of Fig. 5. The tens position of the instruction register controls which program track is to be read from, and the units position controls which of the ten-position instructions on that track is to be read into the instruction register. Once an instruction is in the instruction register, the static contents of this register control the actual information transfer, as outlined above.

Arithmetic operations in this machine are handled in substantially the same manner as information transfer instructions. The accumulator system is a track on the process drum plus the necessary circuitry to handle algebraic addition and subtractions. If the accumulator track is specified as the receiving track in an information transfer instruction, the numeric value of that information is added to the value already on the accumulator track and the result is written on the accumulator track. Actually, the accumulator track may be specified as the receiving track in two ways, add or subtract. In either case the arithmetic system notes the signs of the two numbers passing through it, notes the instruction specification of add or subtract, derives the proper result, and writes this result back on the accumulator track. The circuitry is arranged to treat the 100 positions of the accumulator track as ten individual ten-position accumulators. Each of the ten accumulators has its own sign associated with it. Carries between accumulators are not allowed and an overflow device is included to indicate this condition, should it occur. Transfers out of the accumulator track can either reset the part read out or not, according to the way that the accumulator is specified on the read-out instruction. There is no restriction as to the number of positions that may be read to the accumulator track on a transfer instruction. Thus, several additions may be completed on a single instruction.

As with addition, multiplication is also handled as an information-transfer instruction. There is a multiplicand track, and specification of this as the receiving track will cause the number transmitted to be written on this track ten times. A maximum of nine characters can be
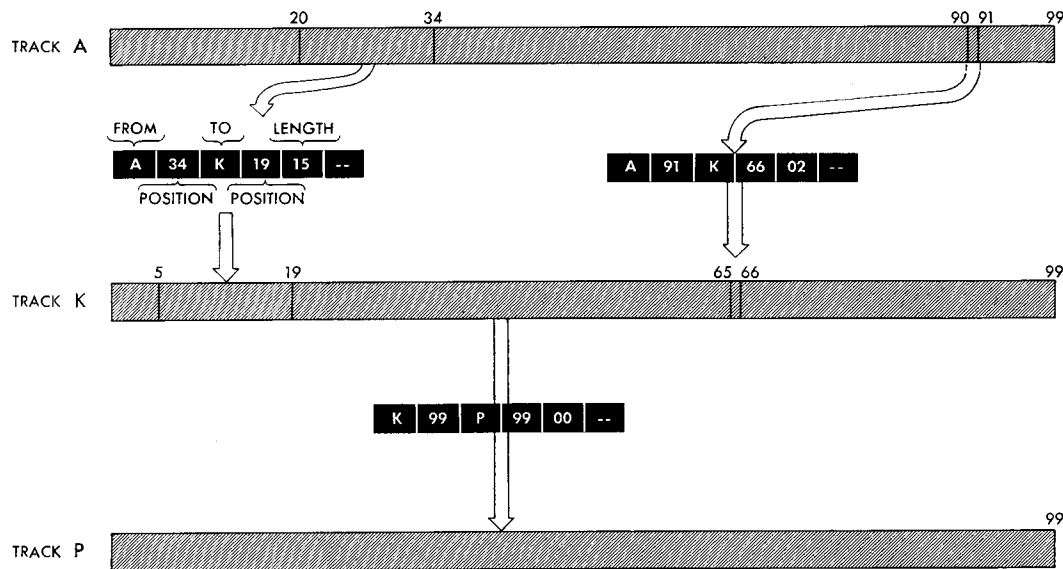
67

Figure 5
Logic diagram for the Random-Access Memory Accounting Machine.

Note: Control circuits indicated are completed by control-panel wiring from exit hubs below to entry hubs above. Decision elements may be used in any desired sequence. Control is returned to stored program by picking appropriate level of program counter.

Figure 6
Information Transfer.

read to this track. Specification of the multiplier as the receiving track in an instruction will cause multiplication of the number transmitted on this instruction by the number on the multiplicand track. The product is stored in the first two accumulators on the accumulator track. The multiplication process is simple, in that each multiplier digit merely chooses when to start adding the information on the multiplicand track into the product register. The multiplicand track may be used for operations other than multiplication. One such use is to send a number there and then add all or part of this track to the accumulator track. In this manner the same number can be used to affect balances in several accumulators in a single instruction. This distribution function is the reason for writing the tenth field on the multiplicand track.

Any information-transfer instruction can be converted from an actual transfer to a comparison of the data specified on the *from* and *to* tracks by placing a 1 in the tenth position of the instruction. The result of the comparison is made available for control operations to be described later. Thus, if we wish to compare a name that is on track A from positions 20 to 44 with a name that is on track J from positions 65 to 89, we would write the following instruction:

A   44   J   89   25   ( )   1

There is also a field compare operation made possible by inserting a 2 in the tenth position of the instruction. This instruction compares the data from the two tracks specified and makes comparison results from each block of 10 characters on the tracks available for control operations to be described later.

In addition to the drum tracks already mentioned,

there are two other units that may be specified as receiving locations in an information-transfer instruction; these are the address register and the character selector. The address register is a five-digit storage unit where numbers are placed to specify the location of the records required from the file. A regular information-transfer instruction is used to place a number in the register. A subsequent operation causes the arm on the file unit to go to the location specified. The character selector is a one-position storage unit to which any character in the system may be sent using an information-transfer instruction. This may be analyzed partially or completely during a control operation within the machine.

This completes the discussion of the information-transfer aspects of the 305 system design. In summary, the objectives were to provide an effective yet readily understandable means for manipulating information in a manner required in business data processing. In general, the 305 has achieved this by:

1. Handling alphabetic and numeric information with equal facility,
2. Providing for variable-length transfers as a basic characteristic of the system,
3. Including an arithmetic system capable of handling several numeric balances simultaneously,
4. Providing an alphanumeric comparison facility.

Processing control

Among the requirements of data processing are a means for altering the way in which different situations are handled. As was pointed out in the first portion of this

69

paper, the 305 uses two mechanisms for accomplishing the required results. A stored program is used for information transfer and a control panel for the control functions. The general approach is to set up the conditions for control with the information transfer instructions and then to wire the means of considering these conditions on a control panel. The ninth character of the instruction provides the means for testing these conditions at a predetermined point in the instruction sequence. An alphabetic character is placed in the ninth position of an instruction, and at the completion of the execution of this instruction the number in the program counter is dropped, and an impulse is made available on the control panel via the hub corresponding to the letter used in the instruction. This hub will be wired through various decision elements back to an entry point in the instruction sequence. Physically, the impulse travels through relay points controlled by the decision elements and picks up a new position in the program counter. The position that is picked up is determined by the status of the decision elements and the control panel wiring.

The decision elements available on the control panel are as follows:
1. Condition of the 10 accumulators—*plus, minus* and *zero.*
2. Contents of the character selector.
3. Status of the latest comparison made.
4. Status, *blank field* or not, of the latest information transfer or field comparisons made.
5. Status of a group of latch selectors (relays) that have been picked or dropped at previous control points.

With these decision elements, the normal process in programming an application is to set up the conditions for several decisions and then to make a multiple branch in the program. The net result of the control system is to provide a mechanism whereby many instructions do not have to be utilized in making the decisions necessary in the process.

This aim is accomplished by concentrating the decisions to a few points on the program. Another feature of the system lies in the programming simplicity achieved, where the programmer can physically put his decisions together on the panel without an elaborate review and rearrangement of the stored program. This latter feature is particularly advantageous, as it is usually in the control area that the more intricate problems in a program develop, and here the programmer has a chance to consider these problems independent of the more voluminous but less intricate information transfers required.

Input-output-file controls

An important part of the 305 design is the operation and control of the input, output, and file units. The three aspects of importance in considering these units are:
1. Means of initiating operations of each unit.
2. Controls within each unit that are substantially independent of the process unit.

3. Interlock system between these units and the process unit.

Operations for the input, output and file units are initiated by control panel wiring.

The controls within each unit must be discussed for each unit. On the card reader two tracks are provided. The card reads directly to one track, while the other track is available to the process unit. A card-feed impulse transfers the track just used by the process unit to card-read status, and the other track from card-read status to availability to the process unit. This impulse also initiates feeding a new card to the track just placed in the card-read status.

The printer and punch both obtain information from a single output track. The control as to what information to print or punch and how, is within the print and punch units. In this manner the process unit only need place the required information on the output track. Print functions such as format arrangement, zero suppression, etc., are specified on the print control panel. Punch arrangement on the card is specified on a punch control panel. The print or punch signal initiates the proper operation. Information on the output track is analyzed to determine the specification as to the mode of printing and punching.

The file is given a new address by transferring a number to the address register which starts the file access mechanism into motion.

A similar interlock system is used for integrating the operations of all peripheral units with the process unit. The basic approach in the interlocking system is to achieve an overlap of operations such as card feeding, processing, printing, punching, and file seeking. This overlap has been achieved by providing sufficient controls for each unit such that interlocks are required only when there is a conflict in the needs of two of the units. An example of such a conflict would be for the process unit to attempt writing on the output track when printing and punching of the data there from the previous operations had not been completed. At such points an interlock would be effected. Writing on the output track would be prevented until such times as printing and punching were complete. Similar interlocks are effected for card reading and other operations where these conflicts can develop.

The result of the interlocking and input-output-file controls is such that for any given application the time chargeable to a transaction on the machine is the longest of the card feed, print, punch or processing functions for that transaction instead of the sum of these. At any given time, it will be normal for the machine to be processing a given transaction, feeding the card for the next transaction, and printing the results of the previous transaction.

Machine speeds

The speeds of the various units within the RAMAC system were set to achieve a balance in the time that

each unit spends upon a given transaction. The printer was the first unit chosen and the speeds in the remainder of the system are balanced to the printer speed. Table 1 gives the speeds of the various units in the RAMAC system.

Table 1 **Machine Speeds**

Unit	Speed
Card feed	125 cards/min
Printer	50 col/sec, 30 cards/min
Punch	100 cards /min
Process unit	30 msec/STEP + 20 msec/STEP with control transfer
File	0.8 sec max, 0.15 sec min

In any given problem the overall machine speed will be limited by one of these units. In loading the file from cards, the machine will normally be card-feed limited, and in punching the file on cards, punch-limited. In a typical billing job it would probably be printer-limited. At two seconds per line we get 1800 lines per hour. At reasonable utilization rates, the objective for the machine of 10,000 line items per day can be achieved.