to instructions and data, and it relieves the arithmetic unit from providing storage and shifting registers. The auxiliary storage is a backup store for the core memory. It provides an economical means to store data and instructions thereby effectively increasing the internal memory capacity and achieving a compromise between cost and access time.

External Control

The over-all system operating philosophy is that of centralized control. Appropriate remote indicators and controls from most of the equipments, including the computer, are provided at the system central consoles. Control of the system is under the direction of a centralized team. To control the system efficiently, and to minimize potential human operating errors, simplified controls and proper operating techniques have been designed. The computer itself has a console designed primarily for maintenance and checking operations, and includes a wide variety of status indicators, as well as the necessary controls to test properly and service the computer.

Fig. 5 shows the RCA BIZMAC computer.

Accuracy Control

The system philosophy of centralized control with its concept of minimum human intervention makes automatic detection and correction highly desirable. The problems involved in providing such a feature are many and complex. Detection circuits are necessary to insure reliable output. Once an error is detected, a correction must be made. The correction is a direct function of where the error occurred in the program, what caused it, whether it was a transient or permanent failure, etc. To simplify this complex problem, errors are classified in two general groups. The first group includes errors resulting from a known permanent component failure or major malfunction of external equipment. The machine is stopped immediately and suitable maintenance is performed or replacement is made. The second group includes those errors that result from either transient or questionable permanent-component failure, or voltage transients. This group generally requires a rerun of a portion of the program to determine if a shutdown is required. All errors in this group are handled in the same manner. A complete rerun of the transaction (computation) in process takes place automatically. The necessary clearing, backing up of tape, and other appropriate operations required are performed prior to the rerun. Programmed counters are provided to limit the number of reruns that can be performed.

Examples of the type of error detection found in group one are as follows:

1. Magnetic tape moving forward when it should be moving in reverse, or vice versa.
2. Magnetic tape circuits not operable.
3. End of magnetic tape.
4. The counter which addresses the high-speed memory during instruction read-out is not cycling properly.

Examples of error detection found in group two are as follows:

1. Parity failures.
2. Adder comparator (arithmetic is performed twice. The second addition is performed using complemented operands and the results are then compared).
3. Verify operation where data are compared bit for bit.
4. Arithmetic overflow.

Conclusion

The RCA BIZMAC computer is a major element of the RCA BIZMAC data-handling system. The several novel features which it incorporates permit it to fulfill its missions in data conversion, data editing, and data generation. It is designed to operate upon variable item and message lengths in order that the system may maintain the economy in reduced tape lengths, and that the computer may achieve maximum useful data rates. Every attention has been given to make it a part of an integrated data-handling system.

Programming a Variable-Word-Length Computer

L. S. BENSKY T. M. HUREWITZ R. A. C. LANE A. S. KRANZLEY

INVESTIGATION of commercial applications for electronic data-processing systems has revealed certain basic characteristics. Very high volumes of input and output data are handled with a modest amount of data calculation. Variability in data length, data occurrence, and in procedures for handling of these data is another major characteristic.

L. S. Bensky, T. M. Hurewitz, R A. C. Lane, and A. S. Kranzley are with the Radio Corporation of America, Camden, N. J.

The manner in which the RCA BIZMAC computer handles variability in all of its aspects has provided a uniquely adaptable tool for commercial applications. The intermediate function of preparing the computer for these applications, programming, is therefore unique in many respects.

Working with clear and concise definitions of commercial applications, the programmer is concerned with applying computer flexibility in an optimum manner. The variable-word-length computer permits concentration of effort in applying flexibility to the handling of data. Programming results may be measured in terms of effectiveness in computer-time and storage utilization, and accuracy control tempered by the availability of well-defined problems and programming time.

Efficiency in Program Composition

One of the prime objectives in the writing of data-processing programs is minimization of the over-all computer time required to accomplish a specific task. In commercial applications, where the work load for the computer is essentially of a cyclic nature, significant cost reductions may be realized from effective equipment utilization. These gains take the form of a smaller complement of equipment, or the performance of more tasks with existing equipment.

It has previously been stated that commercial applications are characterized by a high volume of input messages. Each of these messages is handled individually by the computer until all have been processed. Programs for such applications, then, are used in a repetitive fashion, with each cycle representing action taken on a single message. A shortening of the message cycle by only a few milliseconds will have a significant cumulative effect if the cycle is repeated many times. For example, 36 milliseconds cut from a cycle which is repeated 100,000 times amounts to a total saving of 1 hour of computer time.

A complete and well-organized definition of the problem is the starting point from which an efficient program can be obtained. Characteristics of data handled by the computer (as shown on standard data sheets) directly affect the efficiency of programs. One outstanding example of this is the format of output messages to be printed. A good format will make maximum use of tabulating stops, thus reducing the programmed steps needed for line composition. The columnar alignment of items is another criterion of an output format which may be used to relate the problem definition to efficiency in the program.

Messages entering the computer frequently contain coded items which serve as the basis for decision-making. For example, a 1-digit decimal code may be used to distinguish ten distinct types of transactions. In a computer where instruction modification is easily performed, the choice of values for such codes permits reduction of decision-making sequences of instructions. With latitude in establishing such codes, it is possible to make each value correspond to an address in either the high-speed memory or the auxiliary memory. Decision-making would then consist of modifying a transfer of control (or other instruction) with the particular code value for each message. Other features which facilitate a reduction in the number of instructions executed, but not in the number provided, are consecutiveness, and ordering of codes by relative volume of appearance.

From descriptions of input and output data, the statement of intermediate operations may also be examined for areas affecting efficiency in the program. Inasmuch as the majority of commercial programs are executed in a cyclic fashion, the establishment of volume figures assumes major importance in selecting operation sequences. Emphasis is placed on optimizing those operations which represent major streams of data flow. Conversely, infrequently used sequences may be complex at small cost in over-all computer time.

Unique Characteristics of Variable-Word-Length Programming

Introduction

It has been stated previously that two basic requirements necessary for the production of efficient programs are: (1) a knowledge of the problem; and (2) a knowledge of the equipment to be used in the solution of the problem. Some of the flexibility available to the programmer in the RCA BIZMAC computer is briefly described and illustrated in the remainder of this section. Utilization of the instructions is explained in terms of the application of the RCA BIZMAC equipment to business problems.

A functional subdivision is made into the three broad categories: computer handling of programs, computational aspects, and editing of data.

Computer Handling of Programs

Programs for the RCA BIZMAC computer are stored permanently on magnetic tape. The initial preparation of the magnetic tape is accomplished in exactly the same manner as the initial input of data to the system. That is, a 7-channel paper tape is first prepared and verified, followed by transcription to magnetic tape. The indexed programs are then stored in what may be called a magnetic-tape program file. It is possible to store up to 2,500 different programs on a single reel of tape.

Part of the initial setup of a computer operation consists of transferring the proper sequence of instructions from the magnetic-tape program file to the auxiliary memory. This is accomplished by a short routine which will search the tape for the desired program and make the transfer. Excluding tape search time, it takes approximately 1 minute to load the auxiliary memory completely. Programs that require less than the full capacity of the auxiliary memory take proportionately less insertion time.

For most business applications, the entire program for a particular computer run will be loaded initially. In cases where many irregularities are to be handled automatically as part of one computer operation, those portions of the program used infrequently may be stored on magnetic tape. When necessary, these portions may be entered without any appreciable time delay.

It is quite probable that some applications will require that the program be modified at regular intervals, that running control totals be maintained, that dates be changed, etc. In these cases, a new program tape may be prepared at the conclusion of a computer run. Undoubtedly, changes will be made in existing programs as systems and procedures are altered. Changes of this nature can be made through the use of special service routines designed for this purpose.

Instructions are "surged" from the auxiliary memory into the high-speed memory in groups (multiples of four) of up to 64 instructions. The quantity of data transferred during a surge contributes to the time required to perform the surge function. There are cases where the large surges would be wasteful because only a few instructions are used before transferring control, and other cases where a part of the high-speed memory usually reserved for the storage of instructions is needed for data storage of temporary work area. It is clear that the ability to change the length of the surge automatically during the running of a program is an extremely useful tool for minimizing running time.

An excellent way to reduce program-access time is to refrain from surging. This can be accomplished through effective use of the instructions and data that are already stored in the high-speed memory, by transferring control to instructions that are stored in the memory, and by appropriate modification of instructions.

Address modification, or the ability of a computer to operate on its own instructions in the same manner as it operates on data, is an extremely important characteristic of efficient programming. It is useful where an identical sequence of instructions must be repeated in each case using operands stored in different memory locations. For example, it may be required to advance an address or series of addresses by a count of one preceding each cycle through a "loop." It may be required to advance or decrease an address by any constant amount. In any event, this can be accomplished through the use of the "binary add" or "binary subtract left-justified" instructions.

As has been mentioned, the "binary add" instruction adds RCA BIZMAC characters according to their binary equivalents. For example, excluding parity bits, $K + \$ = (101010)_2 + (000111)_2 = (110001)_2$. Octally, this would be $K + \$ = (52)_8 + (07)_8 = (61)_8$. A bit is carried into the next addition cycle if the sum of two characters is greater than $(77)_8$. Both auxiliary and high-speed

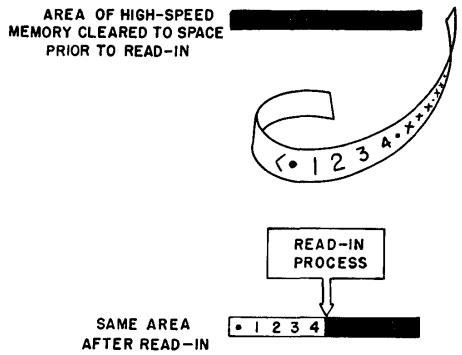

Fig. 1 (left). Status of high-speed memory locations allotted for an item with a maximum length of ten decimal digits

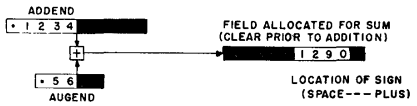

Fig. 2. Addition-operands justified left

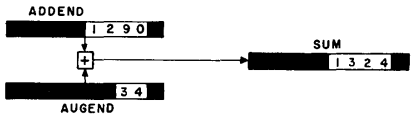

Fig. 3. Addition-operands justified right

memories are addressed octally in the program, and each address is expressed as four octal digits. Therefore, to advance an address by one, two, six, etc., it is necessary to store $(0001)_8$, $(0002)_8$, $(0006)_8$, etc., in the high-speed memory. Each constant uses only three memory locations (including one for the item separator). In cases where unused instruction addresses are available for the storage of the constant, no additional high-speed memory locations are required. For decreasing an address by a constant amount, "binary subtract left-justified" is employed in the same manner as the "binary add" instruction.

It is very convenient in many cases to generate the address instead of modifying an existing address. This can be accomplished quite simply by a "transfer of data" instruction to the surge area.

One application for this technique is in pigeonhole sorting where the numbers to be sorted are converted to addresses which are then placed in a "transfer of data" instruction prior to its execution. The same technique saves considerable program-running time, instruction storage and programming effort, when applied to problems involving the posting of amounts to one of a number of totals. For example, in life insurance accounting it may be necessary to accumulate the premium amounts received by state. The amounts need not be arranged in state order. A code indicating the state total to which each amount is to be posted may be associated with each amount. If the programmer is free to establish the state code for internal processing, a 2-character code may be used that corresponds to the storage location of the state total. The code is now transferred to a single "decimal add" instruction (*A* and *C* or *B* and *C* addresses). Thus, the posting is accomplished without the series of decisions normally required for the accomplishment of similar tasks. Only three instructions are used, including the completion of the addition. If the code is fixed by the procedural requirements it is often possible to establish a transformation procedure for conversion to memory addresses.

In computer handling of programs, a major concern in the use of subroutines is the return to the main routine after execution of the subroutine. Calling in a particular subroutine is no problem since it may be stored at a fixed location on the auxiliary memory. However, since a particular subroutine may be required in several different parts of a program, the return to the main routine necessarily must be variable. The return to the main routine is handled in the RCA BIZMAC computer by storing the address of the point of return in an unused portion of the high-speed memory prior to executing the subroutine. This is conveniently accomplished with the use of the "set up" instruction. The last instruction in the subroutine then must be a "refer" instruction which scans the high-speed memory address containing the point of return to the main routine.

The procedure outlined in the foregoing serves as a convenient tool in the initial development of a program. It represents a scheme whereby a rather complex and lengthy programming task can be subdivided into a number of simpler tasks. These simpler tasks may be programmed and coded individually. Two unused memory locations are assigned to each routine.

Each routine is terminated with a "refer" instruction which scans these locations. This assignment also serves to identify the particular routine. When each of these subroutines has been written, the program may begin with a series of "set up" instructions (one for each routine) in order to tie each of the parts together to form the whole program.

This method is particularly efficient in those types of problems that require these subroutines to be related in a variety of ways, depending on certain decisions and conditions which are established only during the running of the program. In these cases, the proper series of "set up" instructions must follow the decision-making portions of the program occurring within the main routine. Each "set up" instruction controls the "refer" action from one subroutine to the next.

Computational Aspects

An important and fundamental concept of a variable-item-length computer is the complete flexibility available in the definition of items for a given problem.

In an insurance company's billing operation, the policy holder's name and address may be defined as an item. One need only consider the names and addresses of a few friends to see that this particular item could range in length from less than 20 characters to more than 100. On the other hand, to facilitate arrangement of last names in alphabetical order, it may be desirable to define the last name as a separate item. Similarly, identifying information such as stock numbers or policy numbers, with their associated handling codes, may be handled as one item. It should be emphasized that item definition depends primarily on the meaning and intended use of the information.

Since the RCA BIZMAC computer is basically a serial machine and the high-speed memory is used to perform the functions of registers for the storage of all operands and results of arithmetic operations, the number of digits involved in arithmetic operations is unlimited. The use of multiple precision techniques in programming is wholly unnecessary. For example, to program the addition of two positive numbers of 100 decimal digits each requires one instruction that specifies the locations of the least significant digits for each of the operands and for the sum. This applies also in the event the numbers are each one decimal digit in length; or are of unequal length with no restriction on the degree of inequality.

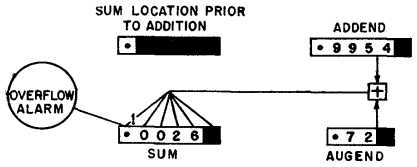

Fig. 4. Overflow in addition operation

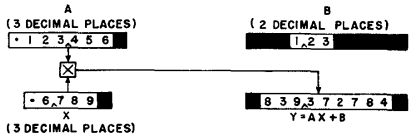

Fig. 5. The computation of y = ax + b involving decimal points

When considering the lengths of the various items to be processed in a particular computer run, it is necessary to determine the maximum number of digits that may occur in each item. Although this is not a requirement for magnetic-tape storage a field consisting of one or more memory locations must be established in the high-speed memory for the storage of each item. This field must be capable of storing the longest of the items for which it was allotted. Thus, if an item is less than maximum length it will be read in to the memory as indicated by Fig. 1. In this case the item is said to be left-justified with reference to its field.

If the item in Fig. 1 is to be added to a second item with a maximum length of five characters, it is necessary only to address the least significant locations of the fields containing the two operands in one "decimal add" instruction. No program routine is required to locate the least significant digits of the items, no instruction to justify the items right is necessary, and no extraction of digits is needed prior to the addition. According to the logic of the RCA BIZMAC computer the characters are read out from each of the fields serially. The actual addition takes place only when two non-space characters are recognized. This is illustrated in Fig. 2.

Gains in programming facility are accompanied by the minimization of the time required for the computer to execute the addition. Time is minimized because the addition is performed on significant digits only, and the time required to search each location for the least significant decimal digit is only half that of the addition of each pair of digits.

Let it be required to add the sum obtained in the above illustration to another 2-digit number where both operands are justified right. This operation is illustrated in Fig. 3.

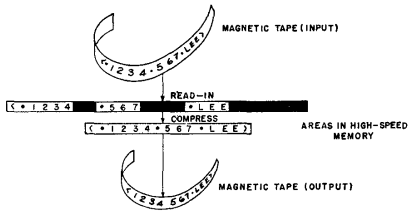

Fig. 6. Compression of output data for maximum tape utilization

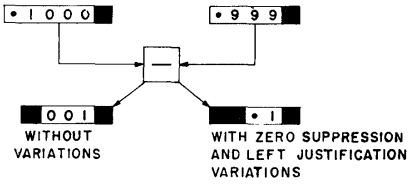

Fig. 7. Illustration of zero suppression with left justification

Spaces to the left of each of the operands add nothing to the sum. In the RCA BIZMAC computer, the addition operation ends on recognition of spaces or item separators to the left of both operands. Only those digits representing the sum are written into the field allocated, and the spaces to the left of the sum are those that were present prior to the addition. "Decimal subtract" and "multiply" are handled in a similar manner making optimum use of variable item lengths.

The ability to multiply and accumulate (i.e., $x = ab + c$) with one instruction is incorporated in the RCA BIZMAC computer. This is accomplished by addressing the product (ab) to a location in which another item is stored (c). This particular feature evolves from the use of high-speed memory locations as partial-product registers in the multiplication process.

Although the number of digits occurring in the results of arithmetic operations is usually predictable, there may be certain instances, particularly in the case of some accumulated totals, where the capacities of the fields allotted for the results may be exceeded (see Fig. 4). In order to avoid such a possibility, a variation of the decimal arithmetic operations is provided to detect overflow. It will actuate an overflow alarm, indicating that a result has exceeded the capacity of its field.

The handling of decimal points in the RCA BIZMAC computer can be considered neither floating-point nor fixed in the strict sense of these terms. It employs the advantage of both and might be termed "absolute variable." All arithmetic operations treat the operands as whole numbers, and decimal points per se are entirely excluded from the operations.

It is the function of the programmer to determine the magnitudes of each operand, and by proper addressing, to insure that the correct results of operations involving decimal places are obtained. In most business applications, this is easily accomplished. In scientific computation this could be accomplished using well known floating-point tecnniques. The computation of $y = ax + b$ involving decimal points is illustrated in Fig. 5 where " $\wedge$ " indicates the theoretical location of the point.

In business data processing there is a definite need for handling alphabetic information in a manner similar to numerical data. This is illustrated quite vividly in the sorting of names into alphabetic order, and determination of the relative order of items such as stock numbers, policy numbers, etc., where the items consist of both alphabetic and numeric information. In addition, considerations relative to address modification (previously explained) require the inclusion of binary operations.

The numbers and letters in the RCA BIZMAC code have been assigned binary codes such that the commonly accepted ordering (i.e., 0, 1, 2,..., 9; A, B, C, ..., Z) corresponds to increasing numerical values of their binary equivalents. Thus, the ability to handle the characters of an item according to their true binary values implies the ability to determine the normal alphanumeric ordering of a pair of digits (e.g. $3A$, CM, AB, etc.).

To determine which of two items containing combinations of alphanumeric characters is of larger magnitude, it is only necessary to subtract one from the other and examine the sign of the results. The "binary subtract" instruction is used. The test of the sign is accomplished with the "conditional transfer of control." Thus, a combination of only two instructions is sufficient to determine the relative magnitudes of two items regardless of lengths and alphanumeric composition.

Editing of Data

Consideration must be given to the form and content of data handled by the various pieces of equipment comprising the system. Messages entering these equipments must have proper control symbols, item ordering, and positioning to insure that output information is correctly produced. The process of adding, deleting, and rearranging data within a message for subsequent operations is defined as editing. Most of the editing work is handled by the computer, since it is the

most flexible unit for data processing.

Many of the problems in editing incoming data for the computer are handled by the "read-in" instruction. It was previously noted that items in an incoming message may be placed randomly in the high-speed memory under the direction of the program. Random composition on the read-in is used to store items in such a manner that subsequent data transfers are minimized. For example, arithmetic instructions require operands to be located in opposite banks of the high-speed memory. When two incoming items are to be added, they are initially placed in opposite banks by the read-in instruction. In the same way, items may be stored in correct order for writing out to tape. The best arrangement of the read-in is made by deciding how each incoming item is to be subsequently used.

Items in a reference file which are not required for a particular computer operation may be discarded by the "read-in" instruction by addressing to a special discard location. In some operations where long, consolidated reference file messages enter the computer for summarization or modification, several hundred high-speed memory locations and clearing operations for these locations may be saved. Item-separator symbols which are required as control symbols for subsequent operations may be generated by providing item addresses in excess of input message requirements. Otherwise, these additional control symbols must be placed in the high-speed memory as part of a special editing subroutine.

Although it is possible to arrange the items of an input message in a form suitable for output, it is frequently the case that more than one arrangement is desired. This happens where several types of output documents and an updated reference file are to be produced by the computer operation. Intelligent use of the rather potent transfer-of-data instruction is called for here.

The amount of processing that can be accomplished during one computer operation is dependent to a large extent on the amount of storage that is available. This refers to the number of tapes that can be utilized as well as internal storage. In the RCA BIZMAC computer, as many as 15 separate magnetic tapes can be connected at the same time. Five of these are input tapes and ten are output tapes. The output tapes may be used in a read or a write status. This multiplicity of input and output tapes makes it possible to prepare many different output documents in a single computer operation, thus eliminating other machine intervention between the computer and the high-speed printer.

Maximum tape utilization is obtained by using the "compress" instruction to eliminate nonessential space symbols from the data to be written out (see Fig. 6). Such space symbols represent the difference between the maximum number of locations allowed for items in the high-speed memory, and the actual number of characters in the items.

Composition of messages routed from computer to printer presents a somewhat different problem. Here, the emphasis is shifted to line and page composition. Left or right columnar alignment of items (justification), item ordering, and the insertion of control symbols are the elements to be considered in programming for a specific printed format. A requirement in business data processing, particularly with regard to output documents, is the suppression of excess zeros in numeric quantities.

The suppression of zeros in the RCA BIZMAC computer is automatic and is included as a variation of the "decimal add" and "decimal subtract" instructions (see Fig. 7). Each item to be written out must be examined for its justification. If an item is left-justified (having the most significant character next to the item-separator symbol) and is to be printed with right justification, a "justify right" instruction must be executed. When items must be left-justified, an arithmetic operation using the justify left variation will provide the necessary shift.

Evaluation of Variable-Word-Length Programming

In programming for commercial data-processing problems, minimization of over-all computer time is the major criterion for evaluation.

Instruction Utilization

The RCA BIZMAC computer provides a list of instructions specifically designed to handle variability in data and variability in processing requirements. An illustration of this versatility is furnished by considering the decimal arithmetic instructions. The foregoing text mentions in detail the following functions:

1. Operands are located and results are stored by one instruction ($A + B = C$), eliminating the need for special registers.
2. No shifting (justification) of operands is necessary since instructions operate on significant characters only.
3. There is no preparatory extraction of characters before operation.
4. The operations are algebraic and sign handling is automatic.
5. Decimal points are program-handled in an absolute variable fashion.
6. Automatic suppression of zeros may be included.
7. Results may be automatically justified.

In addition, the particular features of the computer which reduce instruction requirements for total data-processing functions are important to the programmer. For example, storage of the program in the high-speed memory assists in the use of address modification techniques which are an important requirement of commercial data-processing programs. Random composition of data entering the computer is another feature which makes possible more efficient combinations of instructions in subsequent data processing steps.

Another category of instruction functions is useful for purposes of data-processing economy outside of the computer operation proper. For example, use of the "compress" instruction is related specifically to tape-storage economies. It should be noted this also provides subsequent reduction of read-in time throughout the system.

Storage Utilization

As mentioned previously, the use of variable and adjustable-field item lengths permits appreciable savings. Magnetic-tape space is saved by putting on the tape only these characters that actually appear in the input, thus reducing the length of tape necessary to hold a given piece of information. This implies a reduction in the number of reels of tape and consequently fewer tape-handling devices required. It also results in saving of computer time since tape movement consumes a large portion of the total time. (Savings up to 70 per cent over maximum field data storage are achieved.) Finally, a saving is achieved in the internal storage required by having to provide only sufficient storage for each item to accommodate the maximum length of that item, rather than having to provide the same maximum amount of storage for each item. In the former case, the storage required is merely the total of the maximum item lengths for all of the items to be handled, while in the latter the storage required would be the maximum required for the maximum length item multiplied by the number of items to be handled.

The computer is designed to permit optimum use of internal storage facilities. The use of an auxiliary memory for the

storage of both data and instructions in any desired array, the variability of surge length, the unrestricted ability to transfer control within the high-speed memory, and finally the reduction in high-speed memory work area requirements brought about by the ability to compose output data randomly, exemplify the tools that are in the hands of the programmer to maintain an efficient balance between time and storage.

Adaptability to the Problem

Throughout this paper, pertinent characteristics of commercial data-processing problems have been mentioned when descriptions of functional and programming features called for them. A review of these characteristics in retrospect completes the evaluation of variable-word-length programming by the RCA BIZMAC.

Variability in data (size and occurrence) and in procedural requirements is handled by providing a list of instructions, and other computer characteristics, especially designedf or flexibility. This makes it possible to write very compact programs for handling business problems. Flexibility in assignment of internal storage at several levels is the necessary corollary to the efficient programming of commercial data-processing problems.

From detailed analyses of many types of commercial data-processing problems, it has been found that some categories of procedures are related to the end use of output data. Of greater importance, considerable similarity of procedures among apparently unrelated applications is evident. It turns out that these similar procedures are composed of basic operations which are identical in nature. This simplifies the programming task particularly where equipment functions are specifically designed to respond to problem data-characteristics.

Detailed programming for many commercial data-processing problems will furnish the final evaluation of variable-word-length programming. It is certain that this expansive phase in the art of programming will be marked by further advances in programming techniques; these new techniques in turn will influence future equipment design. Such advances will be based on substantially improved knowledge of data characteristics, the standardization of routine business processes, and the acceptance of mathematical techniques as tools in scientific management.