A Revolutionary Technology

Fig. 6 illustrates and characterizes what is believed will be a revolutionary technology. Cryogenic elements have been chosen for purposes of presentation, but several different types of devices under consideration by research groups would have served equally well. The small squares in the illustration represent logical elements. For easy visualization all the elements shown in Fig. 6 have been magnified. The expanded picture of one element shows as an example, a trigger circuit with its interconnections. The overall dimensions of this trigger are less than one eighth inch by one eighth inch.

Reasoning from this example, the impending revolution in computer technology will be based on:

1. Microminiaturization. Extremely small, active, passive, and interconnecting elements allowing dense packing to meet logical complexity and speed requirements.

2. Batch-bulk processes. As characterized by Professor Buck, it is anticipated that systems will be manufactured by producing interconnected batches of circuits from bulk raw materials in automatically controlled continuous processes.

3. Physical merging of storage, switching and information transmission. Only the merger of logical and systems design with microminiaturization and batch-bulk processes will produce the full impact of the revolution.

This revolution will have far reaching implications in all phases of the computer field, starting with research and proceeding through system design, manufacturing, including the user. It is the purpose of this meeting to bring attention to the impending changes and to provoke discussion.

Computer Design from the Programmer's Viewpoint

W. F. BAUER

MR. RICE HAS given a stimulating introduction to the subject of the impending revolution in computer technology. This paper will continue this introduction by calling attention to the spectrum of activity in computer design. On the one extreme of this design spectrum is the user and on the other extreme is the engineer-designer. Interpolated between these two positions are the following: problem formulator, programmer, system specifier, system designer, logical designer, and circuit designer. Mr. Rice has introduced the subject from the designer end of the spectrum; this paper will make some further remarks of introduction along the lines of the user's viewpoint and then further develop ideas of what the user expects in the way of computer design.

From the standpoint of the user the principal implication of the impending revolution in computer technology is that information processing will cost less; the advances in computer fabrication and in computer system design will mean that a given amount of information processing will cost less or, alternatively, for a given amount of money more information processing can be purchased. This in turn implies a greater sophistry in computer application. Since a major consideration in the application of computers is programming, attention is focused on that activity. Improved computer technology will imply less cost for programming because of the increases in automatic programming sophistication made possible by the machines of more advanced design. In the United States today, a condition of insufficient manpower to program computers efficiently is rapidly being approached; if adequate strides of progress in programming are not made, more and more people with less qualification will be brought into programming, and the costs will rise rapidly as a consequence. The author believes that the limiting factor in computers today is not the switching times of logical circuits nor the access times of memory units. Rather, the limiting factor is the lack of capability, on the part of machine and on the part of the user, for advanced programming and advanced application.

Attempts have been made to pin down the idea of the productivity of the programmer and to determine how this productivity has changed through the years since 1950. One such "programmer productivity index" would be the ratio of programming cost to computer cost. Another such index would be the ratio of the size of computation staffs to the speed of the computer. Any such definition shows a very great growth in programmer productivity since 1950 and, more specifically, shows a growth factor of at least five, and very possibly as much as 25, depending on the definition used and the assumptions accepted. This increase in programmer productivity is due mainly to the computer design improvements which, from the programmer's standpoint, make possible ease of coding, especially through automatic programming techniques. The importance of the computer design from the programmer's point of view is obviously great in the case of the general purpose computer or data processor. It is only somewhat less true in the case of the stored program computer designed for more specific application.

For present purposes some of the factors of modern computer design will be discussed and what may be considered to be the "ultimate" in computer design from the standpoint of the user will be described.

Design Features

There has been much discussion over the past years on trends in instruction repertoires of stored program computers. Here again the subject can be discussed in terms of a spectrum of design possibilities. On the one extreme is the micro-instruction which is not powerful but is universal in character; it provides a small unit building block for the computer program. On the other extreme is the macro-instruction which is more powerful but also more restricted in application.

As instruction repertoires become more "problem-oriented" they become less universal in character and more special purpose. Computer instructions more problem-oriented in nature can be synthesized through automatic programming—or, commands in the user's terminology are translated to the conventional machine instruction. To illustrate how the more problem-oriented machine instruc-

---

W. F. Bauer is with Ramo-Wooldridge Division of Thompson Ramo Wooldridge Inc., Los Angeles, Calif.

This paper is part of a panel session on "The Impending Revolution in Computer Technology."

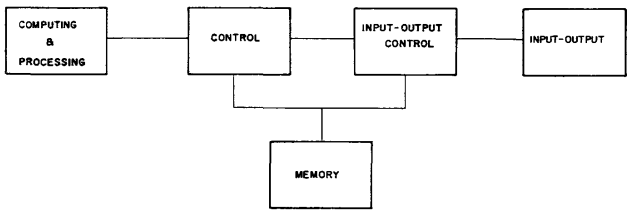

Fig. 1. 1103A-704-705 system schematic

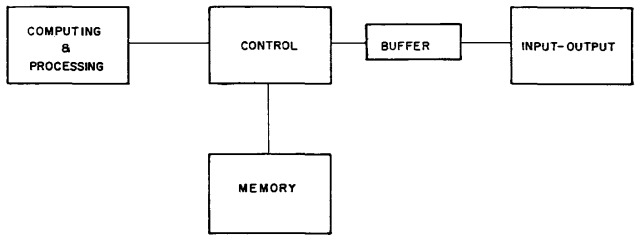

Fig. 2. IBM-709 system schematic

tion repertoire results in its more limited use, consider the very successful FORTRAN algebraic formula translation system device. In nearly all International Business Machines Corporation (IBM) *704* computer installations in the west, less than 30% of the problems use FORTRAN, and very frequently the use is around 10%.

History shows that, in general, special-purpose machines are a poor investment. Consider, for example, the lack of success of computers for the solution of linear algebraic equations. In computer development there has been an obvious trend to universality. Compare, for example, the 16-instruction code of National Bureau of Standards Eastern Automatic Computer (SEAC) with the very large instruction repertoires of the Livermore Atomic Research Computer (LARC) and (STRETCH) computers. However, included in this trend is the march toward including the more powerful instruction, the instruction which is more problem-oriented. To emphasize this, one can recall that in 1951 people who wanted built-in floating point were somewhat hard to find. At that time, many users, and probably more designers, thought that floating-point arithmetic instructions should be programmed rather than built in.

The real solution to the problem of design of instruction repertoires for computers seems to be incisiveness with universality, the set of micro-instructions heavily sprinkled with problem-oriented instructions.

The past few years have seen many important innovations in computer design. Some of these are as follows: the B-box, automatic overflow and underflow handling, the real-time clock available for program interrogation, and the automatic interruption of the computer by asynchronous devices. The last innovation mentioned, "the interrupt feature," bears extra mention and emphasis. The complexity of system design implies that events asynchronous to the computer operation (e.g., card reading, memory interrogation, input from externally timed sources) must be handled by the automatic interruption of the main computing stream; employing the interrupt idea is like asking a staff member to perform a task and then return for another assignment when the first task has been completed. The savings in computer time and programmer time in the employment of the interrupt idea is significant. Happily, this technique, first realized as a standard feature in the Univac Scientific *1103A*, is being included in such new computers as the IBM *7070* and the STRETCH computer.

Another new area arising is that referred to as the associative computer memory or its cousin the indirect address. The associative memory is that first introduced, developed, and used by Newell, Shaw, and Simon[1,2] in their work with a synthesized logic theory machine. Without attempting to describe the technical details of this technique, let it simply be said that it allows the programmer to set up strings of quantities within the computer memory, thereby allowing him to make better use of the computer memory and relieving him of the necessity to specifically outline computer memory requirements on an *a priori* basis. The indirect address technique is developed to a considerable extent on the IBM *709* computer and almost to an ultimate degree on the Gamma *60* computer.[3] This technique essentially allows the programmer the same advantages of the associative memory and, in particular, allows him to deal with names of quantities rather than the quantities themselves.

An idea which may appear important in computer technology from the user's standpoint is the symbolic memory. With this memory, the memory cell itself contains in symbolic form its "address" or, in better terminology, its "name." The programmer, instead of referring to a specific address, refers to a symbolic name in order to summon a quantity from the memory. The assignment of names to specific hardware memory cells is performed by the machine at the beginning of the computation, and a one-to-one correspondence between the symbolic names given and the specific memory cells is effected by the computer hardware. For those who would argue strongly that this is not desirable or feasible, consider the fact that computer programming by means of absolute machine addresses is nearly extinct in the United States today. When a usage becomes this universal, consideration should be given to its inclusion as a hardware item, if this is at all possible.

An interesting phenomenon in computer design is the increasing memory size. Although the memory size of the large-scale computer has increased roughly as the square root of the increased speed, it appears that the advent of concurrency of input-output with computing will change this trend. It also appears that, since large transfers between main memory and auxiliary memory can now be handled so efficiently and economically, there is a limit to, or an optimum for, the size of a high-speed memory. It appears that computer memories in the future will grow with increasing computer speeds but at a rate slower than the square root rate just mentioned.

One of the requirements for larger high-speed memory sizes lies in automatic programming. In the system for the IBM *709*, for example, which allows automatic operation of the computer between and during runs on the computer, a computer control program of 4,000 words in length is kept in the magnetic core at all times. It remains there for the use of the programmer for the handling of untoward situations and for performing the transition from one computer run to the next.

One of the principal faults of the modern computer is that it has characteristics of an arithmetic manipulator and does not have sufficient character along the lines of information handling. The modern large-scale computer is used as much as 40% of the time in program debugging. In this activity the computer is acting almost completely as an information handling device—

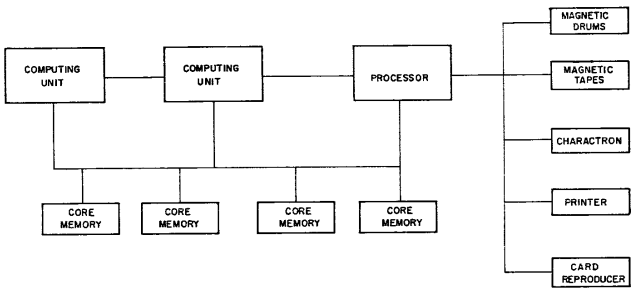

Fig. 3. Univac LARC system

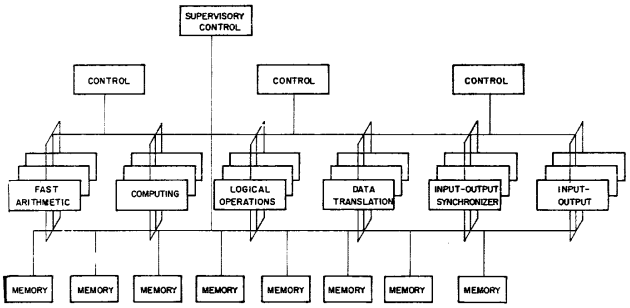

Fig. 4. "Ultradatic" system schematic

as a data processor. If one adds to this percentage the percentage of time the computer performs production operation on problems which are essentially of an information handling or data processing character, one finds that the so-called "scientific computer" in a scientific environment is performing data processing and information handling upwards of 70% of the time. Yet neither instruction repertoire nor system design factors seem to reflect this figure. A striking example is this: A great percentage of time the information processing performed is that of translating alphanumeric input to numerical input, that is, the mapping of the six-bit characters to series of, say, octal digits. Yet almost none of the modern computers allow the programmer ease and facility in the handling of six-bit characters; the translation must always be performed in an indirect way, through synthesis, with extract and shift instructions. The author believes it can be safely stated that no computer in existence today has a system design which reflects the fact that automatic programming is an important and ubiquitous labor saving device.

System Design

System design is the design of the computer with respect to the interrelation of its respective large components such as memory, input-output, control unit, etc. With the advent of the IBM *709* and the Univac *1105* computers, and, more specifically, such computers as the LARC, STRETCH, the Lincoln *TX-2*,[4] and the Gamma *60*, there is a significant revolution in system design taking place. This revolution certainly ranks in importance with the revolution taking place in the design and fabrication of logical elements.

Consider a historical view of design. Four phases of system design can be distinguished: The first of these can be called the "laboratory computer," characterized by such computers as SEAC, National Bureau of Standards Western Automatic Computer (SWAC), and Whirlwind. These computers were strictly sequential in their operation. Input and output interrupted the main computation. Frequently, a megacycle bit rate in the main computation was replaced by a 25 bit-per-second rate for input and output.

The second system design phase could be referred to as the "buffered computer" phase, characterized by the Univac, the IBM *701*, IBM *704*, IBM *705*, and the Univac *1103* computer. With these computers, input-output required only the time necessary for the transfer of information at electronic speeds. With the advent of this phase, input-output was taken out of the domain of mechanical speeds and placed in the domain of electronic speeds. It should be noted that within the computers of this phase there is a wide range of buffering. Fig. 1 shows the simplified block diagram of this computer.

The third phase of system design can be referred to as the "automatic computer" represented by the IBM *709*, the *TX-2* and the Univac Scientific *1105*. Here is seen the first concurrent operation of simple parallelism. With these computers, input-output proceeds independent of the central processing. Fig. 2 shows the simplified schematic of the IBM *709* computer system and, incidentally, shows how the memory is time-shared by the central control unit and the input-output control unit.

The phase which is now being entered could be referred to as the "parallel system" computer phase. This phase is represented by the LARC and the STRETCH computers just appearing. With these computers, there is concurrent operation of large components of the computer. Fig. 3 shows the block diagram of the Univac-LARC computer.

The advent of the "parallel system" computer will have important changes in computer programming. In the parallel machine, in the LARC or the Gamma *60*, for example, where large components of the machine can be time-shared by problems, it may be possible to use the machine to perform automatic programming processing as a fill-in operation during the normal course of the work of the computer. In this case, the use of large amounts of machine time will not inhibit computer automatic programming developments. On the other side of the coin is the fact that exploitation of these machines with parallel operation by automatic programming techniques will be most difficult. (Witness the difficulty of the B-boxes in the development of FORTRAN for the IBM *704*.) For a thorough introduction to the subject of programming of these new style computers, the reader is referred to a recent paper by S. Gill[5].

A failure of the system design of the present computers lies in the fact that the duty cycle of the various components is not sufficiently high. With most computers, for example, the circuits for high-speed multiplication are utilized only a small fraction of the time. In a sense the analog computer system design comes far closer to the ideal. In the analog computer installation, for example, the computer can be spread out and "sized" to the problem. If the average analog computer problem utilizes only a small fraction of the total number of amplifiers and nonlinear units, a high duty cycle can be maintained on all of the equipment. Taking the lead, then, from the analog computer design, it appears that concurrent operations of large parts of the computer should be possible, and that in some way the job for the computer must be made small with respect to the computer. This idea is exploited further in the design of the conjectural computer "Ultradatic" described later.

One of the goals in computer design is

to make the computer act in some sense more like a human. The reader has probably heard of "self-organizing" systems and will undoubtedly hear more of these in the future. As part of the self-organizing aspect, the psychologist or physiologist could refer to the "degree of introspection" of a complicated system in judging how "nearly human" the system is. Certainly the human stops his main stream of activities frequently to reflect on what he is doing from the over-all point of view. This introspection increase can be seen in computer design. In a sense, the interrupt feature mentioned, provides a degree of introspection since one part of the computer system automatically informs another part when it has performed or when it is ready to perform the asynchronous operation. The large-scale computer of the future, by the same token, will periodically examine its own operations and automatically take certain steps to change its function. Ultimately it will change its very nature of character and thereby take on a self-organizing aspect.

"Ultradatic," A Conjectural Computer

A computer system will be described, which is in some sense "ultimate." This is done in the hope that it will stimulate thinking and comment rather than with the thought of foretelling the future. The idea for "Ultradatic" was motivated by the thinking described, concerning the advantages of the analog computer system which can be sized to fit the problem, and by the related idea that it may be possible to make computation more economical by making problems small relative to the computer. It is fostered by the idea that the large computer is more economical to use today than the small computer, or a number of small computers, as long as the total workload is sufficient to keep the large computer busy, or as long as unnecessary expense is not incurred by idle time. The idea further stems from the fact that with increasing frequency one sees computers being operated from remote locations by means of communication over voice channels, or at least extensive input or output of data is being fed to computers from remote locations.

The central idea here is that each large metropolitan area would have one or more of these super computers. The computers would handle a number of problems concurrently. Organizations would have input-output equipment installed on their own premises and would buy time on the computer much the same way that the average household buys power and water from utility companies. In fact, the charges for the services would depend on the type of services rendered. If the problem required extensive use of the higher priced circuits, for example, high-speed multiplying, the rental rate for the time used would be higher than in other cases. The user would be charged for that time used and only that time used, and the accounting procedure would reflect the customers' detailed use.

Fig. 4 shows the schematic of the Ultradatic system. On the figure there can be seen two levels of control, the supervisory control and the normal control unit, and a number of processing units. There is a fast arithmetic unit to perform ultra high-speed arithmetic, and there is a computing unit to provide the lower speed arithmetic with, however, greater possibility for flexibility. There is a logical operations unit which specializes in activities such as sorting and collating. The data-translation processing unit specializes in translation and editing, and the input-output scheduling unit performs detailed control over input and output.

The reader will notice great similarity in Ultradatic to the Gamma *60* computer which was first generally known in this country in the spring of 1958. (The Ultradatic idea was first described in talks given by the author to the San Diego and Rio Grande chapters of the Association for Computing Machinery in late 1957 and early 1958.)

In this large computer system the supervisory control units schedule the multiple problems, monitor the usage of the control units, and initiate the work on the problems by the control units. Each problem has a priority connected with it, and the supervisory control monitors and changes the priorities according to the speed with which problems are being performed. In more advanced configurations the supervisory control would perform some kind of introspection and, perhaps, would effect the change of character of some of the units to other kinds of units depending on the requirements of the over-all problem situation.

The control units interpret macro-instructions, effect the transfer of subroutine variables and parameters to and from the processing units, distribute work among like processing units, and initiate the action of the processing units.

Each processing unit is a small computer in its own right. Each has registers for variables and parameters, each has a limited number of instructions which it can perform peculiar to its particular function, and each processing unit records the time spent on the various problems with which the computer is dealing.

A problem number and a priority number accompany each command sent from the control units to the data processing units. The problem number is essentially an account number used for accounting purposes and the priority number describes the order in which commands are to be performed by the processing unit in case waiting lines develop. Because branch A of a problem may be completed before branch B and the results of branch B are required to continue with branch A, a scheme based on the sequence of operations within a particular problem would be required for interruption or delay of operations.

At first thought one might conclude that the scheduling problem on Ultradatic would be an impossible one; that is, it would be impossible to schedule the various components so that a problem can be done in a short length of time or in a reasonable length of time. It is believed that scheduling a problem would not arise with such a computer. On the basis of the customer's desires a certain initial priority number would be assigned to a problem, and a rough estimate would be made by the computer itself with regard to its workload and as to how much time would be required to perform the problem in the problem multiplexed environment. As the time grew nearer for the completion of the problem, the computer would upgrade the priority if it appeared unlikely that under the current operation the schedule would be met. Thus, the computer by means of the supervisory control would monitor itself automatically and would automatically change the priorities of problems.

A very important characteristic of the Ultradatic computer is that it is modular in construction, and large components can be added or subtracted to make up a configuration which is currently in demand at the particular installation. The processing units, shown in multiples of three in Fig. 4, would be available in any combination and in any practical numbers. If, for example, the records showed that the fast arithmetic unit had a continual backlog, a new fast arithmetic unit would be added to the system configuration. Or, if one of the logical operations units was found to be idle a good fraction of the time, one of these units

could be removed from the system. Thus the computer can grow and change to meet the changing requirements of the installation and the high duty cycle will be maintained at all times.

The memory units of Ultradatic warrant certain attention and discussion. This ultimate computer would have a symbolic memory; that is, the symbolic address of the cell assigned arbitrarily by the programmer would be stored within the memory cell, thus allowing the programmer always to refer to that cell in symbolic address notation. Because of the multiple branches of the programs proceeding at times and at speeds unpredictable, a different technique must evolve for the use of the memory in this computer. With present computers in rocket trajectory calculations, for example, only one computer cell is reserved for the height of the rocket. In this computer system a cell would be reserved for the height of the rocket at each quantized level of height. This would be necessary since the height at 9,000 feet might be required in one branch of the computations while the previously computed height, 8,500 feet, was being used simultaneously in a completely different branch of the computations. The symbolic address notation, now considered part of the address hardware, would include indices or subscripts to differentiate the various values of a variable. When the programmer realizes that the height at 8,500 feet is no longer necessary anywhere in the computations, he returns the cell to the available unused memory pool. There the memory cell can be assigned to another problem or to another value in the same problem. Here the associative memory idea would come into play; a string of unused memory cells would be established, with each cell of the unused memory storing the address of still another unused memory cell, so that the string could be added to or subtracted from with convenience as required.

There are certain everyday problems which would be performed by Ultradatic. For example, once or twice per day, the computer would examine all of the programs in its backlog and would make estimates of a gross character as to the length of running time. Once or twice per day the computer would carry out a detailed accounting and compute bills for its multitude of users. Of course, the computer would spend a considerable amount of its time in compiling and translating.

Existence of computers like Ultradatic would not obviate the necessity for small computers, either general or special purpose. However, almost every installation with a small computer today has at least one problem which should be placed on a higher speed machine. Usually the big problem is run on the small machine inefficiently and uneconomically because of the inconveniences of renting large scale computer time.

With respect to special-purpose computers, most of the ideas of Ultradatic are still applicable if, of course, attention is restricted to those of stored program design. Consider the idea of growth or of the inevitable changing character of the application. A computer truly modular in system design can be increased in power to handle the control job for the new polymerization unit or the new cracking tower. Or, as a new formulation of the control process evolves, the computer can be changed to efficiently handle the new computations. Also, ideas of system design to make programming easier will become more important with special-purpose machines. When real-time control computers appear in abundance certainly one of the limiting factors in application will be the preparation of computer programs.

Although Ultradatic will probably never appear at the computer market place, some of its design ideas may bear fruit. If not, service to computer technology advancement is still provided in unerringly pointing the direction not to follow.

References

1. Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristic, **A. Newell, J. C. Shaw, H. A. Simon.** *Proceedings*, Western Joint Computer Conference, Feb. 1957, pp. 218–39.

2. The Logic Theory Machine, **A. Newell, H. A. Simon.** *Transactions*, Professional Group on Information Theory, Institute of Radio Engineers, New York, N. Y., vol. IT-2, no. 3, Sept. 1956, pp. 61–79.

3. France's Gamma 60, **P. Dreyfus.** *Datamation*, May–June 1958.

4. The Lincoln TX-2 Computer Development, **W. A. Clark.** *Proceedings*, Western Joint Computer Conference, Feb. 1957.

5. Parallel Programming, **S. Gill.** *The Computer Journal*, British Computer Society, 1957.

6. Modern Large-Scale Computer System Design, **W. F. Bauer.** *Computers and Automation*, Jan. 1957.

Discussion

I. O. Naughton (Westgate Laboratory): What do you see as an upper bound (a) to the ability of a computer to modify its program based on partial results? (b) to the comprehensiveness of the instructions that can be provided for a programmer?

Dr. Bauer: The upper bound is the limit of humans communicating with a device to tell it how to modify its program or logical structure or to tell it how to tell itself how to modify its own program or logical structure.

Concerning the comprehensiveness of instructions, there is no upper bound as far as the programmer is concerned as long as the instructions are grouped and presented in such a way so that subsets of them can be used. However, there is an upper bound from the viewpoint of the computer seller or the computer buyer. As the comprehensiveness goes up, the costs, of course, go up also, and the point of diminishing returns is soon met. Tomorrow's technology will allow comprehensiveness through or by means of self-organization.

Jerry Mendelson (National Cash Register Company): Ten years ago we built machines with 10-microsecond adders which we embedded in control configurations which reduced the potential addition (computation in general) rate from 100,000 per second to 10,000 per second. We then provided the control system with a command list in which about one command in ten did useful work, the other nine being required to get ready for this work. Result: 1,000 useful commands per second.

For 10 years we have been smug and complacent about having produced man's most inefficient machine, (1% efficiency), and all we have done is shove this inefficient design up the frequency spectrum in order to obtain faster effective output, 1 microsecond adders replacing 10 microsecond adders. We appear to be heading into the millimicrosecond domain with no great effort being applied to changing the internal design characteristics which drag us down by a factor of 100. It's about time we paid some attention to this area, is it not?

Dr. Bauer: It certainly is time. It seems that too much time has been given to the more local problem of circuit speed in comparison to the time given to the global problem of the effect of an executed instruction or an executed set of instructions on solving the problem to which the computer is being applied.

Robert Norman (Sperry Gyroscope Company): A centralized computer facility such as you envisage, with problem solutions available effectively at people's telephones, runs right into a problem in human relations.

Much business, personal and corporate, involves privacy. The availability of large volumes of data on the private and corporate affairs of the inhabitants of a community constitutes a threat to their privacy. It seems this would result in a degeneration of the centralized computer facility to one only used for the solution of an unimportant portion of the total volume of computation in the community.

Dr. Bauer: This would undoubtedly be a problem with a system like Ultradatic. However, difficulties are not insurmountable. For example, the computer itself would monitor in detail all communications between it and a customer to insure that no information was given to unauthorized destinations. Furthermore, proprietary information could be secured by encoding techniques similar to those used in military communications today. Even today we tend to use the telephone as if it were a

completely secured and private instrument, and it seems that our "shared" computer of the future would be at least that private.

E. L. Harder (Westinghouse Electric Corporation): What is the present unit cost per computation (you define it), and what reduction do you anticipate as a result of this revolution?

Dr. Bauer: This is a very tough question to answer. Everybody can define the cost per operation of the digital computer itself. However, it is very difficult to define the amount of programming that is done in the application, assign some meaningful cost to it, and assign some meaningful effect to the amount of programming that is done.

As a matter of fact I believe that the emphasis on computer design from the standpoint of speed, size, cost, etc., and the relative lack of emphasis on computer design from the programmer's standpoint stems exactly from the fact that it is easy to put a dollar amount on the computer costs but very difficult to put dollar amounts on programming costs. Much attention is being focused on making the computer economical, but nobody is worried about how much money is being spent on programming.

The "programmer productivity" will probably again increase by many factors during the next 8 years. Weighing against this is the higher salaries which are being paid to programmers in reflection of their contribution to the general technology. Because of these facts I would guess that the cost of computation, all things being considered, certainly will decrease but at a slower rate during the 8 years than it has during the past 8 years.

New Logical and Systems Concepts

R. K. RICHARDS

BEFORE making any attempt to outline the course of future development in computer systems technology, it is well to review briefly the present state of the art and how it arrived at that state. The system designs of the first digital computers such as the Harvard Mark I, the Electronic Numerical Integrator and Calculator (Eniac), and others were largely influenced by ideas presented 100 years earlier by Babbage. In general, the system was comprised of a set of decimal registers, each capable of transmitting numbers in parallel fashion to and from other registers. When a register received a number from another source it was capable of adding that number to any number already contained in that register. Computations were achieved by transmitting numbers back and forth among the various registers with, of course, suitable refinements to obtain the desired results.

Although the concept of long sequences of arithmetic operations with storage of intermediate results was an integral part of the earlier systems design, the stored program concept as it is known today was absent. Further, it is quite likely that even if the stored program concept had been developed at the time of construction of the first computers, it would not have been used. The reason is that the registers used for storing numbers were (and still are) bulky and expensive things, either in the electromechanical or electronic versions. It did not seem feasible to think in terms of more than about 100 such registers in a machine, and this amount of storage is not sufficient to utilize the stored program concept in an effective manner.

The advances in computer systems development occasioned by the introduction of the stored program were achieved only because there was a simultaneous development of relatively low-cost large-capacity digital storage components. Magnetic drums, mercury delay lines, and various forms of electrostatic storage were among the first storage media to be used for this purpose although magnetic cores now play the leading storage role.

No similar progress in reducing the cost of logical components was realized however. Even today, the cost of an elementary "and" function, for example, is generally considered to be many times as costly as a bit of storage in a large-capacity storage array. As a result of this situation, the path of systems development has been in the direction of using the minimum possible number of arithmetic and control elements. The result has been the appearance of the one-operation-at-time general-purpose computer with that one operation being as simple as tolerable to the user of the computer. The binary number system appeared at this point in computer development. Originally the sole purpose (and still the major purpose) of binary numbers was to reduce the number of logical components required to make a computer. The considerable burden placed on the programmer as a result of binary numbers was considered of secondary importance.

With the passage of time the decimal system has been appearing in a larger percentage of new computer designs, and the operations capable of execution by the newer computers are more complex and powerful. However, each step of progress in this direction causes an agonizing increase in the cost of the control portion of the computer and is, therefore, taken only after it is well substantiated that there will be more than an offsetting gain to be obtained through factors such as an increase in computing speed or a decrease in programming effort.

The Course for the Future

Other authors have already discussed the possibility of obtaining low cost logical elements as well as low-cost storage elements through the use of film techniques. In view of the considerable advances in computer systems design that were made possible through the introduction of low cost storage elements, it might be expected that corresponding advances might be made through the introduction of low cost logical elements. It is the purpose of this presentation to examine the factors involved in an effort to determine what these advances might be.

It has been observed that whenever an improvement is made that might be described as being of revolutionary proportions, it has been as a result of solving a problem in one field through the application of a body of knowledge from another field. For example, no great improvements in illumination came as a result of studying flames, tallows, or other aspects of candles. Instead, a knowledge of electricity was used in the development of the electric lamp to replace the candle. Similarly, in the field of transportation only limited gains were realized by improving the care and feeding of horses or by developing better breeds of horses. Instead, the revolutionary advances came from a study of mechanical engines.

In the case of electronic digital computers, if anyone were to propose an ultimately successful concept that re-

R. K. RICHARDS is a Consulting Engineer at Wappinger Falls, N. Y.

This paper is part of a panel session on "The Impending Revolution in Computer Technology."