L-84

Journal of Applied Physics

Volume 21, Number 4 April, 1950

High Speed Digital Computers

An Elementary Survey of Present Developments and Future Trends

LOUIS N. RIDENOUR

University of Illinois, Urbana, Illinois

ONE of the most interesting, and possibly one of the most significant, of the post-war technical developments is the great effort that is currently being expended, both here and abroad, on the design, construction, and improvement of high speed automatic digital computing machines. In accordance with the present pattern of support for technical enterprises, the largest part of the effort on such computers is being financed by the federal government; private industry is investing substantial sums; and, somewhat curiously, the universities of the country do not appear to be supporting with their own resources more than a tiny fraction of the work.

Mechanical aids to computation are an old story, of course. But the high speed, automatic, digital, computer—when each of these qualifying adjectives is taken properly into account—is a very new development. Its practical realization cannot be said to have been achieved earlier than the completion of the celebrated ENIAC in 1945. Of this machine we shall have more to say later. Let us pause now to distinguish from other computing devices the type of machine that will occupy our attention here.

First of all, we shall deal with *digital* machines. This excludes a large class of calculating devices customarily lumped under the name of *analog* machines.[1] An analog machine is one which translates into some physical quantity each of the numbers entering a computation; that is, the physical quantity is the measure of the number. The analog machine then manipulates these physical quantities in accordance with the nature of the computation it is desired to carry out. Finally, the physical quantity resulting from such manipulation is measured—that is, turned back into a number—to obtain the numerical answer for the computation. A slide-rule is an analog machine. Numbers are represented, on the slide-rule, by lengths which are proportional to the logarithms of such numbers. To perform a multiplication, the lengths corresponding to the two factors are added, and the resulting total length corresponds to the product, being in fact proportional to the logarithm of the product. This total length is turned back into a number by means of the scales provided on the rule, and the calculation is finished.

A digital machine also uses physical representations for numbers, of course; but instead of using a single physical quantity as the *entire* representation of the number, and thus depending upon a highly accurate presentation, manipulation, and reproduction of the magnitude of this single physical quantity, a digital machine represents by a separate physical quantity each separate digit of a number. This permits a very much greater latitude in design for the digital machine; to handle numbers of ten digits, no part of a digital machine needs to be built with a precision of one part in 10^{10} An analog machine, on the other hand, has no better over-all accuracy than that conferred on it by the precision with which it is built.

Analog machines have been brought to a high stage of perfection and usefulness. Beginning about 1931, Vannevar Bush and his co-workers at M.I.T. developed the celebrated differential analyzer, which was a mechanical device in its first embodiment. During and since the war, electrical differential analyzers of various designs have been constructed, and some of

[1] For a description of analog computers and their uses, see D. R. Hartree, *Calculating Instruments and Machines* (University of Illinois Press, Urbana, 1949), Chapters 2, 3, and 4.

these machines are currently available on the commercial market. We shall not concern ourselves here, however, with such machines. We shall speak only of *digital* computers.

In fact, we shall speak only of *automatic* digital computers. The meaning of this qualification is that we shall not deal with digital machines of the sort represented by the familiar adding machine, or the Monroe or Marchant type of desk calculator. Such machines are very old in conception and in realization, of course.[2] The first practical adding machine based on the use of number wheels was devised by Blaise Pascal in 1642; a machine capable of multiplication (by repeated addition accompanied by suitable shifts) was designed by Leibnitz in 1671 and built in 1694. In 1820 Charles Thomas, a Frenchman, designed the first commercially successful machine; with minor modifications and improvements, this machine was being built in Paris right up to the Second World War.

We shall not even concern ourselves here with the familiar punched-card machines manufactured by the International Business Machines Company. These machines stem from the work of Herman Hollerith, an employee of the U. S. Census Bureau, who in 1889 conceived the idea of using, for entering numbers into a machine and for reading out the results, holes punched in a card. A similar scheme had long been used for controlling the weaving of complicated fabrics on the Jacquard loom, and its application to computing machines had been proposed two generations earlier by Charles Babbage, of whom we say more later.

Desk calculators and IBM machines are excluded from the present discussion because such a machine performs only one elementary computation at a time, and must be instructed anew before it can perform another. That is, we can enter on a desk machine the two factors entering a multiplication, and then rely on the machine to carry out the operation and display to us the product. But, before another multiplication—or indeed any other operation—is performed, we must ourselves inform the machine what numbers enter the new computation, and which operation is to be performed on these numbers.

In an automatic machine, on the other hand, sufficient instructions are given the machine at the beginning, and the machine has sufficient capabilities within itself, to permit the whole of a complicated sequence of manipulations on many numbers to be performed by the machine without human intervention, once the signal to start has been given. Thus an automatic machine is basically different from a desk calculator; it is more nearly a mechanized simulacrum of the whole complex: human computer plus desk machine plus tables of functions for reference plus work sheet on which the course of the computation is sketched o and intermediate results can be recorded. The idea i such an automatic machine is more than a century o Charles Babbage, who was Lucasian Professor of Matl matics at Cambridge, conceived such a device—t Analytical Engine—and began to build it (on Briti government money) in 1835.[3] The Analytical Engi was never completed, largely because the mechanic engineering of Babbage's day was not equal to the ta of realizing his complicated designs, but its plan w astonishingly modern in terms of the ideas which st rule the automatic computer field today.

One more qualifier is left. We shall speak here *high speed* automatic digital computers. Earlier th the ENIAC, at least two types of automatic digit computers were designed and built. Professor Howa Aiken, of Harvard, built with the help of the IB Company his so-called Mark I machine which was electrically controlled, mechanically operated, aut matic computer that was hailed as "Babbage's drea come true."[4] About the same time, at Bell Telepho Laboratories, a machine was built which qualified as automatic digital computer according to the prese definitions, and used electromechanical relays as el mentary computing and storage elements.[5] Both the machines were of limited speed because of their d pendence on mechanical elements whose motion w necessarily slow in comparison with the actions electrical circuits. In the ENIAC,[6] on the other han the operations within the machine are conducted e tirely in terms of the circulation through electrical lin of electrical pulses, generated at a base rate of 100,0 per second, and in terms of the registration of t effects of such pulses by means of electronic "flip-flo circuits which can rest stably in either of two statio states, yet be transferred from one state to the ot in a fraction of a microsecond by a pulse of the corr sign.

The time required in the ENIAC to add together t numbers, each of ten decimal digits, is 200 micros onds—200 millionths of a second. In terms of the tin required by machines now under construction, this long; addition times of about 10 microseconds common in modern designs. But, in terms of the ope tion of machines having mechanical elements, this very fast indeed.

In the remarks to follow, then, whenever the w "machine" is used, it will be understood to refer t high speed automatic digital computing machine. I us now turn to some general considerations on the ov all design of such machines.

[2] For a historical account of the development of desk computing machines, see the article "Calculating machines" in the *Encyclopedia Britannica*.

[3] H. P. Babbage, *Babbage's Calculating Engines* (Spon, I London, 1889).

[4] *Annals of the Harvard Computation Laboratory* (Harvard versity Press, Cambridge, 1946), Vol. I.

[5] S. B. Williams, Bell Lab. Record **25**, 49 (1947).

[6] H. H. Goldstine and A. Goldstine, Math. Tables and Ai Computation, **2**, 97 (1946); D. R. Hartree, Nature **158**, 500 (1

2058

GENERAL REMARKS ON MACHINE DESIGN

While any given machine must be designed as a whole, it is often useful to distinguish as separate various parts of the machine which have various functions. The more important of such parts are:

First, the *arithmetic unit*. This device performs the individual arithmetical operations as required by the schedule of the computation being conducted. It is a very close electronic parallel to the familiar desk computing machine.

Second, the *inner* or *high speed memory unit*. This, which is presently the least satisfactory part of machines in existence or under design, is a device for registering numbers in a permanent way—either the numbers entering the problem or intermediate results—and keeping them accessible for use on demand as the computation proceeds. Any number in the memory must be available on demand in a very few microseconds. The memory is ordinarily so arranged that it can also store *orders*, which are instructions to the machine that govern the course of the computation.

Third, the *control unit*. This part of the machine is in charge of the whole operation. It keeps track of the calculation, determines which individual operation should be performed next, and causes its execution.

Finally, an *input-output unit*. This is necessary to enable the machine to communicate with its human masters. Through this unit, the machine is supplied initially with the data and instructions entering the problem; through this unit, the machine can also read out intermediate or final results. Often, the input-output unit is used to provide a large-capacity low speed storage (on magnetic wire or punched type) for numbers and orders. This outer memory supplements the inner memory, since the latter is seldom as capacious as one would desire.

When a mathematical problem is to be presented to such a machine, it must be programmed; that is, broken down into the individual steps to be performed successively. The numbers entering the problem must be supplied, and so must the *orders*, or instructions to the machine for handling those numbers. The input unit reads the program of numbers and orders into the memory unit of the machine at the beginning, and the work thereafter proceeds entirely within the machine.

A typical order might be translated: "Take the number in stored in memory location 1100 and multiply it by the number stored in memory location 2016. Put the product in memory location 1234 (erasing what stands there now) and then go to memory location 1161 for the next order." Orders are expressed in a code which gives them the appearance of numbers, so far as the machine is concerned. This has two advantages: each memory location can store either a number or an order, indifferently, and arithmetical operations can be performed on the pseudonumbers which are orders, thus changing one order into another when this is required as the computation progresses.

The example of an order just given calls for a multiplication. Similar orders govern addition, subtraction, multiplication, and sometimes certain other arithmetical operations. Another type of order, whose use greatly extends the scope of the machine, is the so-called "*conditional transfer*" order. This provides the machine with a sort of judgment, and enables the further course of a computation to be determined by the results to date, without the explicit intervention of a human operator. A typical conditional transfer order might say: "Compare the result of the last manipulation with the number stored in memory location 1648; if it is larger than that number, or equal to it, go to memory location 1174 for the next order; otherwise, go to memory location 1176 for the next order." Such decisions are a prominent feature of the control of a complicated calculation by a human computer using conventional methods.

Numbers are represented within the machine by groups of non-linear elements each of which has two stable states: on or off. For electronic machines of the present day, the non-linear elements are ordinarily the so-called "flip-flop" circuits derived from the multivibrator invented in 1919 by Eccles and Jordan.[7] Two tubes are so coupled, in a flip-flop circuit, that when one tube is conducting, the grid of the other tube is biased far beyond cut-off. If a signal is received in such a sense as to swing positive the grid of the cut-off tube, then the current in this tube rises and at the same time the grid of the other tube is driven negative. In a fraction of a microsecond, this action culminates in a stable situation in which the tube that was formerly cut off is carrying full current, while the grid of the tube that was formerly conducting is biased far beyond cut-off. Thus the system has two distinguishable stable states, and can by a pulse be transferred from one of these states to the other.

The flip-flop is not the only device which realizes practically this requirement of having two stable states which we may call on or off. We shall see later that there are other, and sometimes simpler and cheaper, ways of achieving this end. Indeed, it is in this direction that we can expect much of the short-term progress in computing-machine design to lie.

Since numbers are represented by a succession of elements which are either on or off, it seems natural to express numbers, for the purposes of the machine, in terms of the number system based on 2, rather than in the familiar decimal system based on 10. In the binary system—the base-2 system—each digit of a number has either the value 0 or 1, and the digits appearing in neighboring places with reference to the binary point (analogous to the decimal point in the base-10 system) differ in value by a factor 2, rather than by a factor 10

[7] W. H. Eccles and F. W. Jordan, Radio Review I, 143 (1919).

as they do in the decimal system. Each number is then represented most economically, by a series of on-off elements, one for each digit of the number. The representation of decimal numbers in terms of on-off elements is somewhat less economical, since it requires the use, to represent each digit, of a collection of on-off elements so arranged that the collection has ten distinguishable stable states. These ten states are then used to represent the values, from 0 to 9, which the digit can assume.

Most of the machines presently under construction use the binary system for internal purposes; this necessitates a transformation of numbers from decimal representation to binary representation at the start and the end of each calculation, but the machine itself can perform this transformation, and the time required for this conversion is small in comparison to the time needed for the internal manipulations of numbers by a machine during the solution of a problem.

There is one other argument for the use of the binary system. Binary arithmetic is the arithmetic of logic, in a universe where a proposition is either true (one) or false (zero). The machine can, then, if it uses the binary representation of numbers, conduct its logical operations in the same formal terms as are used for its arithmetical manipulations.

Let us now pause for a moment to discuss what is called "balance" in the design of a computing machine. This term refers to the simple idea that the most efficient operation will be achieved when the times required to perform the various operations conducted by the machines are related to one another in a reasonable way. To take an absurd example, the requirement that the machine be able to perform whole sequences of computations automatically arises from its internal speed. There would be no sense in performing an addition in 200 microseconds (as the ENIAC does) or in 10 microseconds (as some of the new machines will do), if a slow-moving human operator had, at each stage, to copy off the result and then to enter the next two numbers to be summed.

At a more sophisticated level, we must consider the balance of operating times within the machine. This requires a careful weighing of the relative times required for access to the high speed memory, for addition, for transfer of a number from one place to another in the machine, and for the repeated operations involved in multiplication and division. To a certain extent, decisions on the balance of internal operating times will be made on the basis of the type of problem for which the machine will mainly be used; thus, for example, a machine whose main task was to be addition or comparison of numbers would desirably have a different internal balance of operating times from that appropriate to a machine intended for processes involving many multiplications. But the balance of a general-purpose computing machine can also be specified, and attention to this design consideration is important.

It has already been stated that numbers are ordinarily expressed in the machine in terms of their binary representation. There are two quite different ways of handling such numbers, both of which are in use in machines now building. One is called the serial method, the other the parallel method.[8] In serial operation, a number is transmitted from one place to another in the machine a single digit at a time, until the entire number is spelled out. In parallel operation, each digit of a number is transmitted on its own separate line, and all the digits of a given number are sent simultaneously.

As usual, each of these schemes has its advantages and its disadvantages. The parallel representation is inherently faster, since the entire number is transmitted at once, while in the serial scheme a number is sent one digit at a time. On the other hand, the amount of equipment needed in a parallel machine is greater than that required for a serial machine, since in the former as many lines are needed for transmitting a given number as there are digits in the number, while in serial operation all digits are sent successively down a single line. In a serial machine, exact timing is usually quite significant, since, for example, the successive digits of a number are distinguished from one another by their time of occurrence. In a parallel machine, on the other hand, timing is unimportant, since the place of a digit in a number is distinguished completely by the place at which it occurs in the machine. It is too early to say which of these schemes is the better absolutely, or the better under certain circumstances. They seem today to be competitors of approximately equal promise, and only further work and further development can determine which is the better.

The detailed design of the inner, high speed, memory has a substantial influence on the choice of serial or parallel operation for a machine. The requirements on the inner memory are very simple but very challenging. It must be able to store at least hundreds, and preferably thousands, of separate numbers or orders, and to do this in such a way that each is permanent and may be left stored for long periods without error, while at the same time it is accessible for transmission and use within a very few millionths of a second.

The earliest machines, such as the ENIAC, used for an inner memory banks of flip-flop circuits, so that at least one pair of triodes was required to store each digit of each number in the inner memory. This is such an extravagant scheme that not very many numbers could be so stored; the ENIAC has in its inner memory a capacity of only 20 numbers or orders.

It was clear that greater capacity of the inner memory is needed for achieving proper balance in a general purpose computing machine. This problem has occupied machine designers for some time, and it still occupies them. The three types of inner memory in use are first, a rapidly rotating magnetic drum on which the

[8] D. R. Hartree (reference 1, Section 5.1).

4 of 8 2-84

gits of numbers can be stored as patterns of mag-tization; second, a tube containing mercury or other uid down which the signals representing digits of umbers can be sent as pulses of sound; and third, ectrostatic schemes in which the digits of numbers re stored as patterns of charge on the face of an sulator.

It is worth noticing that the magnetic drum and the nic delay line are methods of *dynamic* storage in the ense that the numbers stored circulate continuously uring storage, while the electrostatic scheme is one of *tatic* storage. It has therefore sometimes been said that he dynamic storage methods are best suited to a nachine of serial type, while the static storage fits est a machine of parallel design. This is only partly rue. A dynamic storage method produces one digit fter another, to be sure; but a multiplication of the nits used for dynamic storage (magnetic drums, mer-ury delay tubes) permits each digit of a given number r order to be entered on a separate memory unit, so hat the whole number can be available at once, as it must be in a parallel machine. On the other hand, static methods of storage are not immediately suitable for use in a serial machine, but can be used if the digits of a number are read off from the static storage one after another.

Let us now turn to a consideration of the capabilities, uses, and limitations of computing machines currently under design or construction.

CAPABILITIES, USES, AND LIMITATIONS OF MACHINES

The nature of the orders presented to a machine has already been mentioned. What we must understand at this juncture is that the number of orders a machine can appreciate and act upon—its vocabulary—is built into it by its designer. An analysis of the operations of arithmetic discloses that the operations of addition, subtraction, multiplication, and division can be defined in terms of the elementary operations "join" and "meet." But the formalism necessary for this reduction is so complicated that it is more sensible to instruct the machine in terms of the four conventional operations of arithmetic than it is to reduce (in programming) every arithmetical operation to its basic terms as expressed by the operations "join" and "meet." How far can we profitably go in this direction of complicated arithmetical operation? Is it worth while, for example, to instruct the machine how to take square roots?

The ENIAC is so instructed. On a single order, the ENIAC will take the square root of a designated number. But the operation of extracting a square root is expressible in terms of the four elementary operations of arithmetic, and thus a machine which does not understand a one-word instruction to take a square root can nevertheless extract such a root if it is provided with a sequence of orders designed to produce such a result, by manipulations which the machine does understand.

This is not a trivial point. To increase the vocabulary of a machine—to increase the number of single orders which it understands—greatly complicates the design of the machine. To simplify the machine by reducing its vocabulary greatly complicates the business of programming a problem for the machine. Here, too, a balance must be struck: between machine complexity and program complexity.

In fact, we meet here one of the major differences of philosophy which has so far occurred in the design of machines. It can be illustrated by noticing that the Harvard Mark I machine is provided with a built-in table of four-place logarithms, so that it has the capability to take the logarithm of a number by looking it up in this table. The designers of other machines have asserted that, since reference to a table of function values is one of the least efficient elementary operations performed by a machine, it is better and faster to have the machine calculate for itself any function values needed, using the infinite series expressing the function. The inefficiency of tabular reference arises because either function values are stored in the inner memory (which is already too small for proper balance in most calculations, and therefore ought not to be used up for function tables), or else function values are stored in an outer memory, where consulting them takes a very long time on the extremely rapid time-scale of the machine. The storage of the form of a function makes far smaller total demands on memory space, and the computation of needed values of the function is, in general, far faster than is consultation of the outer memory to find such values. Thus, in this latter philosophy, tables of functions are an anachronism existing today only because not everyone interested in performing calculations yet has access to a high speed automatic digital computing machine.

Of course, the user of a machine often meets a situation in which he must perform repeated calculations of the same kind. If this calculation is not one comprised in the vocabulary of the machine, it must be programmed for the machine in terms of the elementary operations which are encompassed in that vocabulary. Once this is done, then the program which results can be preserved and used again and again. That is, an extended computation will ordinarily consist of a major routine and a number of sub-routines, the latter being of such a nature that they can be used, if necessary, over and over again as the main computation progresses. If, for example, the machine is not designed to understand an instruction directing it to take the square root of a number, then the square-root operation can be programmed in terms of the four basic operations of arithmetic. If that program is saved, it can be used over and over again whenever the major computation demands that a square root be taken. As a machine is used for extended periods, the number of such sub-routines which are built up and filed away for future use will be extended; and it is not too much to expect that programming will thereby be greatly simplified.

It is of some interest to compare a high speed automatic digital computing machine to the human brain, and such comparisons have often been made in the past.[9] Such a comparison must be very carefully made. From what we know now, the central nervous system of animals, including men, is composed of individual neurons or nerve cells which are on-or-off devices like the flip-flop. However, man's central nervous system has ten thousand million such elements, while the most complicated computer so far built has only about ten thousand. This factor of a million which separates computing machines from brains is most important. For one thing, it permits the brain to perform its operations with a very considerable redundancy, so that a failure in a single element of a single chain of neurons is not important in terms of the result.

Present day machines, on the other hand, operate in a unitary way. The machine has so few "neurons" that it must use each as if it were infallible. This puts a fantastic premium on reliability in a computing machine. Von Neumann has estimated that, if a machine of the present complexity and design is not to commit errors more frequently than one per four hours, on the average, then its individual parts must have such a reliability that one failure occurs per million million elementary operations. This is some thousand times better reliability than has so far been achieved in conventional telephone practice, where one failure occurs, on the average, per thousand million operations. One of the unanswered questions about computing machines is whether such a reliability can be achieved.

This question has another side. We must ask whether such a reliability needs to be achieved. Are there not methods of checking which can, at every juncture, protect a machine from the consequences of error?

The answer is that there are. Claude Shannon, in his important paper on the "Mathematical theory of communication," has pointed the way to a method of checking which can correct automatically the errors that may be committed by a computer.[10] The scheme involved, which is due to Hamming, requires, to be sure, that other numbers beside those involved in the computation be handled and transmitted, but ensures that failures in the machine will not affect the results of a computation. In effect, the use of such a scheme introduces redundancy for the sake of reliability, and thus moves the logical design of computing machines a step nearer the logical design of the brain. The advantage of Hamming's method is that it provides a prescription for introducing the needed redundancy in the most economical and efficient way, as can be proved by Shannon's communication theory.

Probably one of the most promising directions of development now visible is this matter of evading intolerable requirements for reliability by suitably chosen means of checking. Various rather simple-minded schemes for checking have been proposed in the past—for example, to put the same problem simultaneously on two independent machines, and then to compare the results—but the checking method mentioned above is the first to have a sound theoretical basis.

Let us return to our comparison between computing machines and the brain. The individual on-off elements of a computing machine operate about a thousand times faster than do the neurons of the central nervous system. This constitutes the principal superiority of computing machines to brains; in almost every other respect the computing machines which we now can build are incomparably inferior to the nervous systems which we all possess.

We have already noticed the greatly superior complexity of the brain. Warren McCulloch, a professor of psychiatry at the College of Medicine of the University of Illinois, has observed that the complexity of the ENIAC, the most complicated machine yet built, is about that of the nervous system of the flatworm.

In terms of space and power, there is no comparison. Your brain is contained in your skull, and it dissipates less than twenty-five watts even when it is in full activity. The ENIAC, a million times less complicated, fills a large room and uses 120 kilowatts of power. Twenty kilowatts more are needed to run the blowers which keep it cool.

The great disparity between a machine and a brain must be taken into account in preparing a problem for solution on a machine. In the words of Lady Lovelace, the daughter of the poet Byron (she was speaking of Babbage's Analytical Engine): "The machine can do only what we know how to order it to perform."[11] It can obey instructions exactly, but it must have instructions or it will engage in an absurdity. This means that every possible eventuality of a complicated calculation must be visualized by the human computer when he programs the problem for the machine, and each such eventuality must be provided for in terms of explicit instructions.

Professor Hartree calls this "taking the machine's-eye view" of a computation, and he cites an instance of his own failure to do so.[12] He had failed to foresee that the argument in a trial solution might become negative since he required finally a positive argument. In the absence of explicit instructions on this point, the machine "did its best, and this was something quite sensible according to its structure and the limited instructions which *had* been given to it, but quite different from the correct small extrapolation of those instru

[9] Warren S. McCulloch, Elec. Eng. **68**, 492 (1949).
[10] C. E. Shannon and Warren Weaver, *The Mathematical Theory of Communication* (University of Illinois Press, Urbana, 1949), Section 17, p. 48.
[11] H. P. Babbage (reference 3, p. 44).
[12] D. R. Hartree (reference 1, Section 7.9, p. 92.

6 of 8 L-84

ions which a human computor would make as a matter of course."

Up to this point we have emphasized, and perhaps overemphasized, the degree to which a computing machine is dependent upon its detailed design and on the orders it receives. The impression has probably been given that these determine entirely the operation of the machine. All this is quite true, under ordinary circumstances, but there is another way of using a machine which is worthy of remark.

I refer to the so-called "Monte Carlo" type of computation.[13] This method replaces the exact analysis of a complicated problem with a sort of statistical experimentation. That is, under properly chosen safeguards to ensure randomness, solutions of the problem at hand are secured for many special cases, and the results are examined to determine the relative probability of various over-all occurrences.

This will be clearer in terms of a specific example. The Monte Carlo technique has been applied to the problem of neutron diffusion in a scattering and absorbing medium. Neutrons of a known energy distribution fall on such a medium in a beam of known intensity; at 10 cm depth (say) what will be the neutron density and velocity distribution? At each encounter with a nucleus, a neutron has a certain chance of being captured, of being elastically scattered in one direction or another, or of being inelastically scattered with energy loss of some amount or other. The probabilities of all these events vary with the energy of the incident neutron and are different for different target nuclei (for the absorber may and usually will contain more than one nuclear type. Rather than seeking an analytical solution for this very complex problem, Ulam[13] has followed the detailed history of many individual neutrons through the absorber, at each encounter with a nucleus determining the outcome by casting dice (in effect), the dice being suitably loaded to reproduce the probability structure appropriate to the event concerned. When this has been done for [illegible] many individual neutrons, the statistics of their resultant behavior presumably approaches that which would result from a full-dress analytical solution of the problem.

A high speed machine is admirably adapted to apply this method, for its speed enables it to obtain a statistically significant volume of results in a reasonably short time. However, if a machine is to do this, it must be instructed how to gamble. This is accomplished by providing to the machine, at each juncture where gambling is required, a random number whose size determines the result of the play. In fact, schemes exist for the generation by analytical processes—and therefore by and within the machine itself—of "pseudorandom" numbers which have all the necessary properties of randomness. D. H. Lehmer[14] has proposed a particularly elegant and useful scheme of this sort. The pseudorandom numbers produced according to Lehmer's prescription are easy to generate in a machine, and meet all of the standard tests for randomness. After one has made such tests, it comes as a distinct shock to find that all these numbers are divisible by 17.

There are good indications that the Monte Carlo method may be one of the best ways for solving partial differential equations on a machine. Certainly its power and scope seem destined to grow.

FUTURE TRENDS IN COMPUTING MACHINES[15]

There are three important questions which concern the future of computing machinery which I should like to discuss. The first is: "Who is likely to possess large high speed computing machines in the future?" Some workers in the computing-machine field seem quite pessimistic about the ultimate wide availability of such machines, on the grounds that they are complicated, expensive, and difficult to keep in order. This is a point of view with which I entirely disagree. I fully expect that a competent high speed computer will very soon be regarded as an important and inevitable part of the research equipment of any university having even the most modest research pretentions.

Thus, the computing machine is to be viewed not as being in the category of the large astronomical telescope, which is a pleasant but optional luxury for a university, but rather as being in the category of the electronuclear particle accelerator, which is a necessity for any university desiring to cultivate modern nuclear physics. It is a trifle surprising that so few American universities have accepted this view; in fact, only the University of Illinois is presently building a machine on its own funds for its own purposes.

Of course, the wide availability and the general acceptance of high speed computing machines will be greatly forwarded by improvements in reliability and reductions in cost. Such developments can be expected to arise from the continued developments in components and the improvement in the logical design of machines.

Let us now ask: "How large, how fast, and how complicated should a large, high speed, general-purpose computing machine be?" It seems unlikely that machines more involved than the ENIAC will ever be built. More recent machine designs are more ambitious in terms of speed of operation, in terms of the size of inner memory, and in terms of the general competence of the machine. In spite of this, such new machines have fewer tubes than the ENIAC; they use these tubes harder, so to speak. I think that the answer to the question of where to draw the line in designing a general-purpose computing machine is set entirely by con-

[13] Ulam, Symposium on Large-Scale Digital Calculating Machinery, Harvard University (September, 1949).

[14] D. H. Lehmer, Symposium on Large-Scale Digital Calculating Machinery, Harvard University (September, 1949).

[15] L. N. Ridenour, Symposium on Large-Scale Digital Calculating Machinery, Harvard University (September, 1949).

siderations of reliability. That is, a large general-purpose machine ought to be as big and complicated and competent as it can be made, subject to the limitation that it must not commit an error oftener than once every few hours. There is no other significant limitation on the total complexity of the device; for the machines which are now in design or construction are still quite inadequate to deal with many problems which we should like to put to them.

One simple example is contained in a remark of Hartree, quoted to me by Professor Aiken. Hartree said that the fastest computing machine available today is still too slow by a factor of 10^{10} to solve the problem of the wave equation for the copper atom. Similarly, the people who are designing the air traffic control plan for the country are proposing to put computers in key locations on the airways of the country, to handle the problems involved in air traffic control. Preliminary estimates of the scope of these problems have indicated that present-day machines will be severely taxed to deal even with the volume of air traffic that exists today.

We want, therefore, to make computing machines as large and as complicated as we can, for the most ambitious machine which we can realize today is powerless in the face of problems that we can readily pose, but not yet solve.

This brings us to my final question: "How can a computing machine be made more reliable, so that its complexity can be increased without increasing the chance of failure?" Much of the answer to this question will depend on the further development and the elaboration of the checking schemes which have already been referred to. However, the obvious way to increase the over-all reliability of present-day machines is to look toward as complete as possible an elimination from such machines of vacuum tubes and electromechanical relays. These two components are presently the major sources of failure in existing machines, partly because they are so numerous, and partly because they wear out with continued use. We need computing elements that can perform the same non-linear functions as those we now achieve with tubes or relays, but elements which are far less prone to depreciation in use.

The vacuum tube also has the disadvantage that it requires large amounts of standby power. McCulloch has remarked that, if a machine built on present principles were to have as many neurons as does the brain, it would require a skyscraper to house it, the power of Niagara to light its tubes, and the full flow of water over the Falls to keep it cool. There are now visible several developments which promise to compete with the vacuum tube for this and other purposes. First, semiconductor devices of the type of the transistor seem to have great promise. Second, magnetic devices like those reported by the Harvard group at the September symposium hold out the hope that they can serve in the place of tubes, with much greater reliability and without requiring power.[16] Finally, the pioneering work of Bowman[17] on electrochemical elements for computers offers a substantial promise in this same direction.

Probably there is no more rewarding direction for computer development than to explore the possibilities of using unconventional elements in the design of such machines. If the vacuum tube can be replaced with a cheaper, simpler, and more durable device, then the over-all competence of practicable computers can be greatly increased. Such a development will bring in its train a greater availability for computers of the present sort, and a wider general use of computing machines of all kinds.

[16] Way Dong Woo, Symposium on Large-Scale Digital Calculating Machinery, Harvard University (September, 1949).

[17] J. R. Bowman, Symposium on Large-Scale Digital Calculating Machinery, Harvard University (September, 1949).

L - 84