Diagnostic Programs for the Illiac*

DAVID J. WHEELER†, AND JAMES E. ROBERTSON†, ASSOCIATE, IRE

Summary—**The diagnostic programs used for maintenance of the ILLIAC, the University of Illinois' digital computer, are described. The uses of diagnostic programs for fault detection, fault isolation, and periodic computer servicing are discussed. The characteristics of the "leapfrog" program, both as a detection program and as an isolation program, are described in detail. Descriptions of one of the more complex isolation programs and of a typical servicing program are given. Pertinent characteristics of the ILLIAC and techniques of fault isolation are also included.**

Introduction

THE MAINTENANCE of an electronic digital computer presents unusual problems for the engineer.[1] A computer is a complex collection of elementary circuits. Although the repair of any individual circuit is simple, the location of the particular circuit at fault among the hundreds of faultless circuits poses a problem of major proportions.

Furthermore, the standard of reliability required is an order of magnitude greater than for other electronic apparatus. Fortunately for the engineer, the computer itself can be used as a versatile test instrument for the localization of faults.

In this paper we discuss first the pertinent characteristics and principles of operation of the Illiac. Next, we describe the typical faults which occur and the effects they have on computer operation. Finally, we discuss the use of three types of diagnostic and servicing programs which enable us to use the computer to diagnose its own troubles. These three kinds of programs answer the questions: Is the computer working correctly? Which part of the computer is at fault? How should this analogue control be adjusted?

Because persistent faults can usually be traced easily with a voltmeter, this paper is concerned mainly with intermittent faults. Refined methods are often required for intermittent faults, especially when the error rate is small.

* Decimal classification: 321.375.2. Original manuscript received by the Institute April 27, 1953.

† University of Illinois, Engineering Research Laboratory, Urbana, Illinois.

[1] Other papers discussing the maintenance of digital computers were published in the 1953 Convention Record of the I.R.E.

TABLE I

Characteristics of the Illiac

Computer type		parallel, asynchronous, general purpose
Register capacity		40 binary digits
Memory capacity		1,024 words each of 40 binary digits
Number of tubes		
Memory		900
Arithmetic unit		1,100
Control		600
Input-output		100
Total		2,700
Type of instruction		Single address, two instructions per word
Number of digits defining an instruction		8 binary digits
Number of digits defining a memory position		10 binary digits
Operation times		
Multiplication	max.	822 μsec
	min.	642 μsec
Division		772 μsec
Addition		72 μsec
Input		4 msec per character
Output (punch)		40 msec per character
Total operation time		3,000 hours[1] (approx.)
Tube failures (excluding cathode-ray tubes)		120[1] (approx.)

[1] On April 20, 1953.

Characteristics of the Illiac

The Illiac, which was completed in September, 1952, is the second automatic electronic computer built at the University of Illinois. It is of the same general type as the Institute for Advanced Study computer at Princeton.[2] In particular, it is a parallel computer with an electrostatic Williams memory. The memory is the only synchronous part of the computer, the rest of the control being asynchronous and designed so that the completion of one operation initiates the next. The computer works internally in the binary system and has 40 binary digits

[2] Descriptions of digital computers similar to the Illiac are given in: G. E. Estrin, "A description of the electronic computer at the Institute for Advanced Study," *Proc. Assoc. for Computing Machinery*, pp. 95–109, Toronto, Ontario, Canada; September, 1952; and R. E. Meagher, and J. P. Nash, "The ordvac," *Rev. Elec. Digital Computers*, pp. 37–43, 1952. (Proc. of Joint AIEE-IRE Computer Conference; December, 1951).

for a single word or number. The instructions are single-address instructions and packed two in a word. The input unit reads teletype tape by means of a photoelectric tape reader and the output unit punches teletype tape. Some of the Illiac characteristics are given in Table I.

The Illiac Memory

The memory is of the electrostatic Williams type, binary digits being stored as charge distributions on the phosphor of commercially available 3KP1 cathode ray tubes. A digit is read from the memory by sensing the appropriate charged area of the phosphor with the electron beam; the resulting potential change is electrostatically coupled to a wire screen on the outer face of the cathode ray tube and is amplified to the signal level of the logical circuits of the computer.

Such a memory is subject to a variety of faults. First, flaws in the phosphor may make storage marginal or impossible. Second, frequent consultations of one area of the storage surface may affect the digits stored in the immediate vicinity. Third, small noise signals may be generated in cathode-ray tubes or amplifier circuits, causing errors in stored data. We shall refer to these as flaws, read-around faults, and random faults, respectively.

Susceptibility of the memory to error has affected both the physical structure and the circuit design of the Illiac. Unlike the remainder of the machine, a pluggable chassis is associated with each of the forty digital positions of the memory. Three controls for each of forty cathode-ray tubes are readily available for adjustment. A separate test rack is used for preliminary selection of cathode-ray tubes and for fault isolation within a pluggable chassis.

The Arithmetic and Control Unit

Arithmetic and control units of Illiac are a complex arrangement of a few types of direct coupled logical circuits, circuits in which tubes are used in an on-off fashion. These are best described from a functional viewpoint.

In the arithmetic unit, flip-flop registers are used for number storage; gates are used for number transfers from one register to another. Numbers in two registers are added with a parallel logical adder, subtraction being carried out by using a complement. Halving and doubling is done by shifting numbers right or left. Two registers are used to perform a shift by gating from the first to the second and then back to the first with the digits displaced one position to the right or to the left. Multiplication and division are performed as sequences of additions or subtractions with shifts. Since the computer is a parallel one, corresponding circuits for each digit are activated simultaneously.

In order to localize a fault in the arithmetic unit we have to find both the digital position and circuit involved. Although an arithmetic error may be discovered as a single digit error, it does not always follow that it occurred in the indicated digital position as it may have been shifted before it was discovered.

The control circuits supervise the sequencing required in the arithmetic unit, the selection and execution of instructions, and the use of the memory. A particular circuit is identified by its function, and failures are localized by interpretation of the malfunctioning produced.

The sequencing circuits of the control are designed so that the completion of one operation initiates the next. This allows circuits to operate at their natural speeds and causes certain faults to stop the computer.

The Input-Output Unit

The Illiac is equipped with a photoelectric tape reader, a tape punch, and a teletypewriter. These input and output devices perform mechanical operations under the control of electronic impulses supplied by the computer. Faulty operation results when a mechanical part is out of adjustment.

Detection Programs

The maintenance engineer of an electronic digital computer must be able not only to localize faults quickly when they occur, but he must also be able to minimize the chance of faults occurring during scheduled operation time. For the latter purpose, a stringent program which thoroughly tests all parts of the computer is needed. We call such a program a detection program. The detection program is designed to exercise each component of the computer through all its possible states. Furthermore, the duty cycle has to be high enough so that all parts of the computer are tested under dynamic conditions. Certain circuits of the computer are connected to many other circuits. For example, a clear driver is used to clear simultaneously all the digits of a 40 digit arithmetic register to zeros or to ones. To test this circuit, it is not necessary to try all of the 2^{40} combinations of digits, because it is known that the maximum and minimum load conditions occur as the registers are filled with ones and zeros. Thus, using these special cases, it is possible to test these circuits adequately without trying all the many combinations. It will be noted, however, that such circuits need specially devised tests.

It is almost unnecessary to state that the test program should be designed so that an absolute minimum number of errors escapes detection. For example, in testing multiplication, we must test all of the digits of the double length product and in testing division we must verify that the remainder is correct.

We are at present using Leapfrog III as a detection program on the Illiac and will describe it in some detail. It is called the "leapfrog" because it has been arranged to "leap" through the memory so that the entire memory is tested. The leap is such that each word of the leapfrog occupies every position in the memory for about one second. Thus every memory position is subjected to all of the different intensities of use from the various kinds of storage in the program.

The memory is tested by a "comparison test" which

takes place as the leapfrog is moved through the memory. Fig. 1 shows in diagrammatic form how the program is moved through the memory. At any one time, copy 3, which we call the working copy, is using copy 2 as the raw material to manufacture copy 1. As each word of copy 2 is translated to become a word of copy 1 it is also compared with the corresponding word of copy 5. If we trace the history of a particular copy at a given position in the memory, we discover it is manufactured (as copy 1), tested for correctness (as copy 2), used (as copy 3), and again tested (as copy 5). This ensures that the program is always checked before it is used, thus giving the maximum chance that a memory error will be found before it causes the leapfrog to act incorrectly. It also ensures that errors occurring in copies numbered 3, 4, or 5 are also detected.

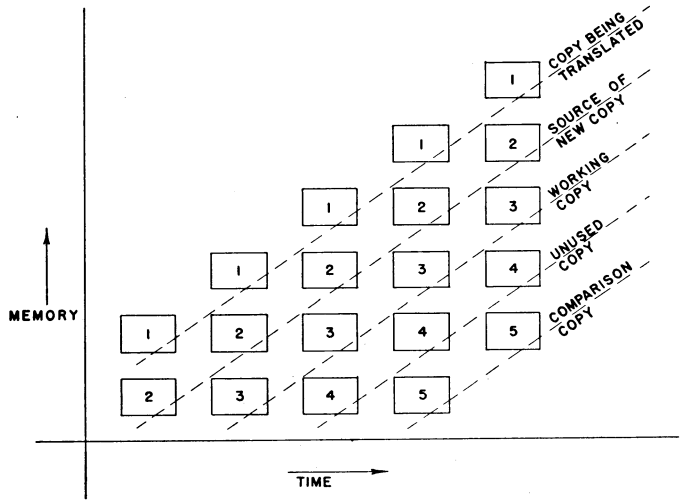

Fig. 1—Motion of the Leapfrog.

The leapfrog contains a stringent arithmetic test. This is split into two parts, a multiplication test and a division test. Both these tests use, and therefore test, other instructions besides multiplication and division. The tests are based upon identities such that all the digits of the numbers involved are checked and any single-digit error will be detected. The numbers used in the arithmetic test are pseudo-random numbers generated from the intermediate results of the previous arithmetic tests. The randomness of these numbers ensures that each digital position of the arithmetic unit is tested under all conditions.

Besides the two tests already mentioned there are additional tests which are performed only once per leap. These additional tests, which are listed in Table II, check certain common circuits of the Illiac under maximum load conditions.

The leapfrog is used to check the serviceability of the Illiac at least twice daily, and is also run during intervals when there is no other demand for computer time. As a result of this policy, and since the leapfrog is more stringent than programs used for calculation, nearly all intermittent faults are first detected by the leapfrog.

TABLE II

INDIVIDUAL TESTS OF THE LEAPFROG*

NAME	EFFECT
Multiplication	A general test of the arithmetic unit, includir the use of multiplication instructions.
Division	A general test of the arithmetic unit, includir the use of division instructions.
Comparison	Compares copy 2 with copy 5 so that memor errors are detected.
Carry test	Tests the full propagation and collapse of tl carry in the adder.
Ones test Zeros test	Tests the functioning of the registers when fu of ones or zeros. This essentially tests commc driver circuits of the arithmetic unit.
Logical order test	This tests the logical instruction. Every digit position is tested in all conditions.
Shift counter test	This tests every digital position of the shi counter and recognition circuits.
Input-output test	This tests the ability of the input-output un to read and punch in all digital positions.
Occasional input-output test	This tests the ability of the input unit to ignor certain characters and read correctly a group c characters, and tests the punch while contin ously punching.

* Note: The first three tests are done 128 times per leap, the ne six tests are done once per leap and the last test once per 128 leaps

FAULT ISOLATION

When a fault has been detected, it is isolated and re paired as quickly as possible. Unfortunately, there is n simple step-by-step procedure which is applicable fc isolation of all types of faults. We shall, however, dis cuss the isolation procedures applicable to a majorit of intermittent failures encountered in operation of th Illiac.

We have noted that the Illiac is composed of thre classes of units; the memory circuits, the mechanic parts of the input and output, and the logical circuit of the control and arithmetic unit. The first step in th resolution of a failure is the isolation of the fault to on of these units. Once this is done, a more detailed analys is needed to define more precisely the location of th fault. The methods of analysis are as varied as the fault

Our fault isolation methods are based upon the fac that nearly all faults are first detected by the leapfro It has been possible to incorporate into the leapfro many of the features of an isolation program withou affecting its stringency as a detection program. In th sections which follow, we describe the isolation feature of the leapfrog, and also the more precise localizatio techniques which are necessary.

The Isolation Properties of the Leapfrog

The first leapfrog test was written for the ORDVAC while it was at the University of Illinois. Although it wa

[1] The ORDVAC was built for Army Ordnance by the Universit of Illinois and has been in operation at the Aberdeen Provir Ground, Maryland, since March, 1952.

a stringent detection test, the ORDVAC leapfrog was unsatisfactory for the isolation of intermittent faults. The reason was that a long time elapsed before some errors were repeated so that it proved desirable to extract as much information as possible from each error as it occurred. The following features of the isolation program have therefore been incorporated in the versions of the leapfrog prepared for the Illiac.

When the comparison test of the leapfrog fails, the corresponding words from copies 2, 3, 4, 5 are printed with their respective memory locations. The intermediate results of the translation are also printed, so that an error of translation can be distinguished from a memory error. If the memory is at fault, the digital position and location of the error can be found from the data.

When the arithmetic test fails, all the intermediate results are printed and the test is automatically repeated. This action of testing and printing continues until the test is satisfied. Thus, if we have an intermittent error, we eventually obtain a correct set of intermediate results. This allows us to determine within one or two instructions the place at which the error occurred. On occasions it has even been possible to determine which step of the multiplication has gone wrong.

Besides these diagnostic features, the arithmetic and the special tests are arranged so that intermediate results used further in the calculation are stored in two memory locations and these are also printed out. This allows us, when one of these tests fails, to say definitely if the memory or arithmetic unit was at fault. When one of the special tests fails, a test identifying number and the intermediate results are printed to enable us to determine the nature of the fault.

The words and numbers are printed out in sexadecimal (base 16) notation, rather than the decimal system, because this is more helpful in diagnosing binary faults. The layout of the printed results has been chosen so that the error is as obvious as possible.

Occasionally a memory fault causes the working copy to become incorrect. In this case a special routine is used to read the leapfrog from the input tape and compare it with the working copy in the memory. The program prints discrepancies so that we can determine the nature and location of the memory fault.

Isolation Procedures

Because the leapfrog is a stringent detection test there are practical limitations to its diagnostic powers. When an error has been detected with the leapfrog further steps are often required to isolate the fault. However, it is relatively simple to find faults in the input, output, or memory circuits from the data supplied by the leapfrog.

Fault Isolation in the Mechanical Parts of the Input-output: Failure of an input-output test of the leapfrog indicates whether the tape reader or punch is at fault. Simple programs which test the faulty mechanism at a higher duty cycle are then used if required. Such failures are generally cured by mechanical adjustment.

Fault Isolation in the Memory: A memory fault is first isolated by the leapfrog to a particular digit of a word in the memory. A cathode-ray oscilloscope is then switched to the chassis of the failing digital position. By inspecting the wave forms displayed, it is usually possible to discover whether the chassis or cathode-ray tube is at fault. If the cathode-ray tube is at fault, the trouble can often be cured by adjusting the controls of the cathode-ray tube; but occasionally replacement of a cathode-ray tube is necessary. Faults which occur in a circuit of a chassis are diagnosed on a separate test rack after the faulty chassis has been replaced by a spare. No attempt is made to isolate a fault within a chassis while it is in the computer.

Fault Isolation in the Control or Arithmetic Unit: Faults in the arithmetic or control unit are usually caused by bad vacuum tubes or faulty connections. Circuits in this part of the Illiac have been conservatively designed, and failures of components other than tubes have not occurred.

Intermittent faults are caused either by shorted tubes, bad solder connections or by marginal circuit operation resulting from tube deterioration. When an intermittent fault is encountered we endeavor to increase the error rate. This can be done by increasing the duty cycle in the suspected part of the computer by a specially designed program written for that purpose. For intermittent shorts the error rate can be increased further by vibrating the suspected part of the computer with a hammer. With the program to give indication of the computer failures, we can discover the element of the computer most sensitive to vibration. If the error is due to marginal operation, then an alteration of a power supply voltage will often cause the error to become persistent, so that it can be traced. If the fault cannot be reduced to a persistent one, then measurements are made in the suspected circuits, either with a voltmeter or with an oscilloscope.

In the arithmetic unit, the technique of interchanging two parallel units is often used to verify other indications during the final localization process.

The effects of faults in the arithmetic unit and in the control are quite different and are usually easy to distinguish. Since each circuit of the control is associated with a control function, a faulty control circuit can be traced by careful observation of the effects of the malfunctioning. In the arithmetic unit, a faulty circuit initially causes a single digit error; the error may, however, be propagated before it is detected in such a way that the circuit at fault is difficult to find.

The computer is used as a versatile test instrument for localization of some intermittent control faults and for nearly all intermittent arithmetic unit faults. Usually the fault is detected by the leapfrog so that some localization clues are available; for example, it may be known that the fault caused an error during a multipli-

cation. For the more difficult faults, a sequence of isolation programs is used, each successive program being shorter and applicable to a smaller part of the computer than the previous one. The programs used in the final localization of the faults are usually very short and are written on the spot, being discarded when the fault is found. Often the program can be reduced to two instructions, using a special mode of operation of the Illiac known as "instruction pairs." In this mode of operation two instructions are set up on a forty digit flip-flop register and obeyed alternately, with continually increasing addresses (modulo 1024).

The Programmed Multiplication Test: The isolation programs used for localization of faults are as varied as the faults themselves. An example of one of the more complicated isolation programs is the programmed multiplication test, designed to localize faults causing errors during a multiplication.

The Illiac performs a multiplication as a sequence of right shifts and additions. During the multiplication, the duty cycle is high in many parts of the arithmetic unit, and some highly intermittent faults occur only during multiplication.

A multiplication can be characterized as a two dimensional array of digits as shown in Fig. 2a. In a parallel machine one of these dimensions is the digital position in the registers and the other is the step of the multiplication (or time).

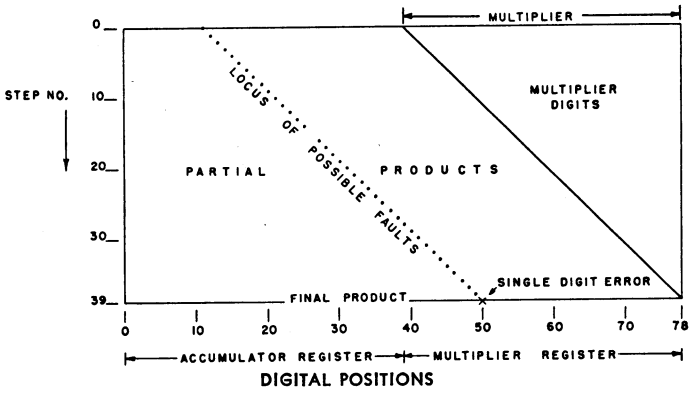

Fig. 2a—Multiplication with single digit error in product.

From a single digit error in a product we can find the relationship between the digital position at which the error may have occurred and the corresponding step. This relationship is represented by the dotted line of Fig. 2a. Unfortunately, not much information is given about the position or step at which the error occurred. The programmed multiplication test is used to isolate such a fault.

The test consists of two parts. The first part is a programmed multiplication and is used to generate in tabular form the two dimensional array of digits. The second part is a series of 39 partial multiplication tests, each of these tests splitting a single multiplication into two partial ones; the first simulates the initial n steps and the second simulates the last $(39-n)$ steps of the multiplication. The tabular values of the programmed multiplication are used for comparison with the final values of the first partial multiplication and are used as initial conditions for the second one.

When a partial multiplication test fails, information is provided for fault localization as indicated in Fig. 2b. An error resulting from a single faulty digital position may be detected in any of 39 digital positions of the product. On the other hand, if the error is detected by a partial multiplication test the range is reduced as shown on the diagram.

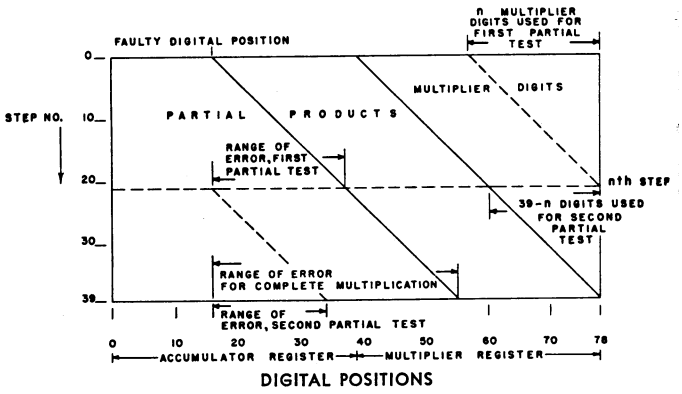

Fig. 2b—Fault localization with partial multiplication tests.

Sometimes an oscilloscope is needed for the final localization of the fault. The oscilloscope presents a cross section of the two dimensional array of digits, that is, a sequence of digits at a particular digital position. This display can be compared with the appropriate column of the table calculated by the programmed multiplication test, thus helping in the final stages of the isolation of the fault.

Computer Maintenance

Occasionally certain controls have to be altered to keep the computer in good adjustment. Such adjustments pertain to the memory, input, or output unit, as the rest of the computer is designed using components in on-off circuits. A typical adjustment is the optimization of the read-around ratio using the intensity control of a cathode ray tube.

To facilitate such adjustments, servicing programs have been written. The programs supply appropriate test conditions to the unit being adjusted and detect malfunctioning. A typical servicing program is the read-around adjustment program. This program continually scans the memory for points of low read-around ratio and indicates these points. If an adjustment is made while this program is running, the effect on the read-around ratio is easily seen, so that the optimum control setting can be discovered by trial and error.

In order to prevent gradually deteriorating components from causing errors during the regular scheduled operating time, marginal tests are performed periodically. There is no marginal testing equipment built

into the Illiac, and the tests are performed by altering the power supply voltages, ac and dc. The leapfrog is used to detect the point at which a circuit fails, the printed error indication being kept as a record. When the tolerance range of any voltage becomes too small for satisfactory performance, the components causing the trouble are localized and replaced.

During the past nine months much more trouble has been caused by shorted tubes and open filaments than slowly deteriorating tubes so that few failures have been prevented in this manner. However, as the Illiac ages, these marginal tests should prove more valuable.

Conclusion

Diagnostic and servicing programs are essential for the efficient maintenance of an automatic digital computer. Since the engineer's knowledge of the computer and the power of the test programs are functions of one another, one would expect the diagnostic and servicing programs to be continually improved, especially with a new computer. It is our opinion that only by frequent and intensive searches for the weak elements, can one maintain the degree of reliability required for a truly serviceable computer, and furthermore, the degree of reliability which can be maintained is determined by the power of the test programs.

Acknowledgment

The material presented in this paper describes the joint efforts of the staff members of the Digital Computer Laboratory of the University of Illinois. The authors are especially indebted to Professor J. P. Nash of the Computer Laboratory for his many helpful suggestions during preparation of the paper.