Technical Memo No. 72
LARC

April 2, 1958
Publications Engineering Dept.
J. Wolfram

AN INTRODUCTION TO THE LARC® DATA-PROCESSING SYSTEM

Information in this memo may not be final. Check with Publications Engineering Dept. for latest revisions.

C O M P A N Y C O N F I D E N T I A L

TABLE OF CONTENTS

APPENDIXES

TABLES

ILLUSTRATIONS

SECTION I

INTRODUCTION

1.1 GENERAL

The LARC® intergrated general-purpose data-processing system is designed by the Remington Rand Univac Division of Sperry Rand Corporation to solve a variety of problems, especially those beyond the range of current data-processing systems. Although the LARC system is expected to be used primarily in the fields of science and engineering, it is also adaptable for the solution of business problems. The system is provided with a complete set of programming aids, including automatic programming.

1.2 COMPUTER AND PROCESSOR

A basic LARC system contains a computer and a processor, each of which has most of the attributes of a general purpose computer but performs somewhat specialized functions in the system. The primary function of the processor is the flexible, parallel, and coordinated control of all input-output operations and transfers between the auxiliary and the main storage. The computer is designed to perform rapid arithmetic computation with a minimum of interference. If increased computing capacity is required, the basic system may be expanded readily to include a second computer. The computers and the processor are controlled by separate programs.

The two computers in an expanded system can be programmed and controlled to solve jointly a single problem or each can solve independently one or more separate problems. The processor is designed to take care of the input-output and auxiliary storage needs of both computers and to do any necessary editing of output data. If input-output demands are not excessive, it may be used to run a sorting, merging or almost any other type of side routine simultaneously with and independent of the computer programs.

1.3 STORAGE

There are four levels of storage in the LARC system which differ in speed, capacity, and costs.

The first level of storage is represented by flexible multipurpose fast registers contained within the computer. These registers are composed of tape-wound magnetic cores having a read-regenerate or clear-write cycle of one microsecond. They are used interchangeably as accumulator registers for storing operands and results in arithmetic operations or as index (B) registers for storing constants used in address indexing operations. Up to 99 12-digit fast registers may be included in each computer.

The second level of storage is a magnetic ferrite-core storage which has a nominal read-regenerate or clear-write cycle of four microseconds. The ferrite-core storage, accessable to both the computer and processor, serves as the main storage of the system and as a common communication link among the computers, the

processor, the auxiliary storage, and the input-output units. To increase the rate at which reference may be made to the main storage, it is divided into independently-operating modules. Since the computers and the processor have access to the same storage, practically any degree of interchange can be achieved among them by alerting one another to the presence of information in a particular part of the storage or by one causing the other to transfer control to a new sequence of instructions in the storage. Up to 97,500 12-digit words of ferrite-core storage may be included in the system.

The third level of storage consists of magnetic drums. The drums are used to replenish the main memory and therefore have a sufficiently high transfer rate and capacity to keep abreast of the unusually high computing rates of the system. Data can be transferred between the main storage and the drums (with a single synchronizer) at a continuous rate of more than 330,000 decimal digits per second. A maximum of six million 12-digit words of drum storage may be included in a system.

The fourth level of storage consists of magnetic tape units which have a data transfer rate of 20,000 alphanumeric characters per second and a virtually unlimited capacity. Although the tape units may be used as relatively long-term storage, they are more often used in the LARC system as fast input-output.

The magnetic drums are controlled by the processor program and, indirectly, by the computer program in much the same way as the input-output equipment. Therefore, to simplify the description of the computer and processor which follows, the drums are classed as input-output although they are actually a type of storage.

1.4 ON-LINE INPUT-OUTPUT

A full complement of both on-line and off-line input-output equipment can be provided with the system. The on-line equipment consists of:

(1) Magnetic tape read-write units for the fast introduction of data into the system and the fast recording of output for subsequent conversion on an auxiliary device or for long-term storage.

(2) An electronic page printer which employs a cathode ray tube for direct, fast recording of output data in either tabular or graphical form. The printer can represent output data as numeric or alphanumeric characters in an edited or unedited format, or as plotted curves complete with callouts, titles, scales, and grid patterns.

(3) Electro-mechanical line printers for multiple-copy printing, a line at a time, of numeric or alphanumeric data in an edited or unedited format.

(4) A card reader for introducing data into the system directly from punched cards.

(5) A console typewriter printer, with an attached paper tape reader and punch, for communication between the computer or processor program and the operator or engineer.

1.5 OFF-LINE AUXILIARY EQUIPMENT

The off-line auxiliary equipment that can be provided with the system includes:

(1) The Univac High-Speed Printer for printing in an edited format data recorded on magnetic tape.

(2) The Unityper II for direct keyboard recording of data on magnetic tape.

(3) The Tape Verifier for direct keyboard recording of data on magnetic tape or verification and correction of data already recorded on magnetic tape.

(4) The Punched Card to Magnetic Tape Converter.

(5) The Magnetic Tape to Punched Card Converter.

(6) The Paper Tape to Magnetic Tape Converter.

(7) The Magnetic Tape to Paper Tape Converter.

1.6 MODULAR CONSTRUCTION

The LARC system consists of modular units, ranging from solid-state component packages to input-output units, storage units, and complex computer units, which can be joined in various numbers and combinations to form a balanced system for a particular range of problems. The types and number of modular units of equipment that can be included in an expanded LARC system and that are included in a typical basic system, are listed in table 1-1. A simplified block diagram of the completely expanded system is shown in figure 1.1. With the exception of the core storage, which must be added to the system in units of four, single units may be added to a system up to the maximum allowed for expansion.

Each cabinet within the system contains its own power supplies, clock pulse generators, and heat exchangers. The synchronizers are modular units of control in the form of solid-state component packages contained within the processor cabinet which has the potential for accepting all of the synchronizers in the expanded system. Identical units of the system are functionally interchangeable and are designed for off line maintenance while the remainder of the system is operational.

1.7 TRAINING AND REFERENCE MATERIAL

The following training and reference material will be available with the LARC system:

(1) Operator Manual - normal procedures for starting and operating the system.

(2) Programming Manual - descriptions of computer and processor instructions, instructions conventions, programming and debugging techinques,

a standard processor program, standard error and contingency routines, and miscellaneous service routines.

(3) Automatic Programming Manual - a description of an automatic program compiler provided with the system and instructions for its application, modification, and expansion.

(4) Maintenance Manual - troubleshooting and maintenance procedures, servicing schedules, and diagnostic routines for the computer, the processor, and the system as a whole. Detailed information for off-line maintenance of input, output, and storage devices is in the individual manual for each device.

(5) Instruction and Operation Analysis Manual - an easily-referenced analysis and brief description of computer and processor instructions and operations. Each instruction and operation is traced through the steps of its execution with a description in shorthand notation of signal functions and timing.

(6) Service Data Manual - a compilation of waveforms, data on voltages, and other service data in easily referenced form.

(7) Circuit Manual - a functional description of standard circuit modules in the computer, processor, and storage and non-standard circuits in the computer and processor.

(8) Logic Manuals - separate manuals for the computer and processor describing their logic circuits and the operations they perform, illustrated with truth tables, block diagrams, timing diagrams, and simplified logic diagrams.

(9) Control Console Manual - a description of the controls and indicators on the operator and engineer control consoles together with an analysis of the effect each control produces within the system, and information on the maintenance of the consoles themselves.

(10) Core Storage Manual - a physical and functional description of the core storage including non-standard circuits (Standard circuits modules are described in the circuit manual for the complete system), information and procedures for off-line testing, troubleshooting, and maintenance.

(11) Input, Output and Drum Storage Manuals - separate manuals for each of the input-output devices and the drum storage. Each manual contains a physical and functional description of the device and information and procedures for off-line testing, troubleshooting, and maintenance.

(12) Parts Lists - an identification of parts for replacement purposes.

(13) Engineering Drawings - wiring diagrams, wiring tables, plug and socket tables, fuse tables, logic diagrams, and signal tables.

Table 1-1

Modular Units of Equipment in a Typical Basic and a Completely Expanded LARC System

Equipment	Basic	Expanded
Magnetic core storage units (2500 words each)	8	39
Computers	1	2
Fast registers (multipurpose)	26	99
Processor	1	1
Drum-read synchronizers	2	3
Drum-write synchronizers	1	2
Tape read-write synchronizers	2	4
Electronic page printer synchronizer	0	1
Line printer synchronizers	1	2
Card-reader synchronizer	0	1
Console printer synchronizers	1	1
Magnetic drum storage units (250,000 words each)	12	24
Uniservo II magnetic tape units	4	40
Electronic page printers	0	2
Line printers	1	2
Card Reader	0	1
Operator control consoles	1	2
Keyboards (numeric)	1	2
Console printers (alphanumeric)	1	2
Engineer control console	1	1
Operator control panel	1	2
Keyboards (numeric)	1	2
Console printers (alphanumeric)	1	2
Computer engineer control panel	1	2
Processor engineer control panel	1	1

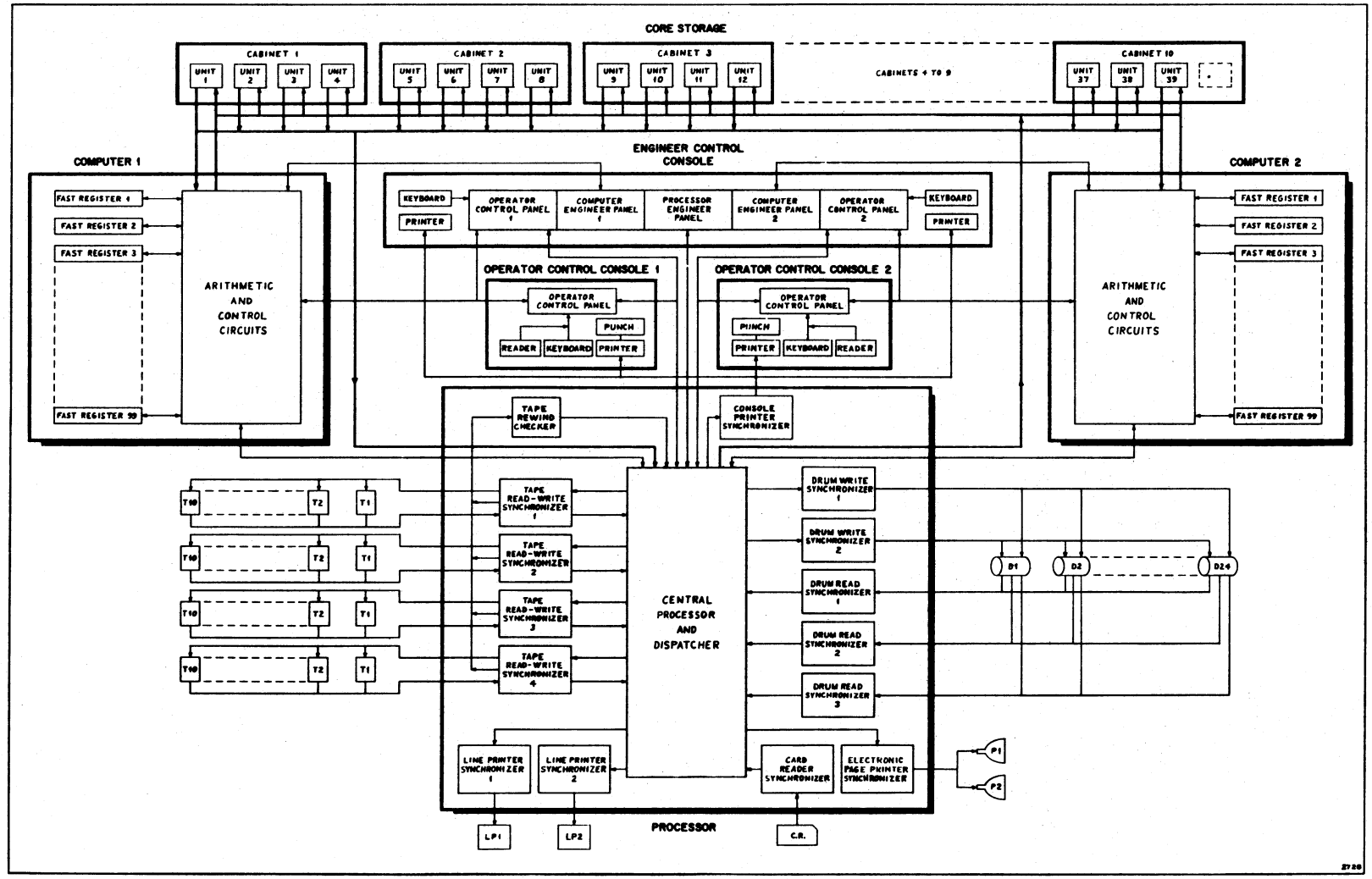


Figure 1-1. Simplified Block Diagram of a Completely Expanded LARC Data-Processing System

SECTION II

COMPUTER

2.1 GENERAL

The computer is designed to rapidly and economically perform fixed or floating point arithmetic operations in single or double precision. To accomplish this, as many operations as possible are performed automatically and in parallel. Means are provided to automatically: modify references to the storage by the instructions, control program loops, trace programs, and monitor machine and program errors. Not only are the bits and digits of a computer word transferred and operated upon in parallel but practically all secondary operations such as input, output, storage transfer, address modification, floating point, and error checking, which might take time from the cardinal arithmetic operations, are performed in parallel with the arithmetic operations.

2.2 INSTRUCTIONS

The computer has a repertory of some 75 instructions. Many of the instructions are in the nature of variants on other instructions. Their inclusion provides the programmer (or more likely the compiler) with a wide choice of alternatives to suit the requirements of a particular problem and increases the speed of computation by enabling operations to be performed with fewer instructions than would otherwise be required. To facilitate programming, the instructions have been made as straightforward as possible. Any oddities in the instructions or illogical restrictions on their use have been avoided.

Many of the instructions are in effect, small built-in subroutines. By designing into the instructions operations which formerly had to be programmed, much more of the computer program and the efforts of the programmer can be devoted to furthering the computations at hand rather than to organizing the computations or the program itself.

Performance times for representive arithmetic instructions of the computer are given in table 2-1. The instruction times listed are all inclusive and include the time required for storage access, address modification, error checking, etc. Also, all input, output, and auxiliary storage operations may be assumed to be performed in parallel with the instructions. For a complete list of computer instructions and descriptions of computer word formats refer to appendix A. A description of the character codes used in the LARC system is given in appendix B.

Table 2-1. Representative Instruction Times in Microseconds

Instruction Type	Add or Subtract	Multiply	Divide
Single Precision Fixed Point (11 decimal digits and a sign)	4	8	32
Single Precision Floating Point (9 decimal digits, an exponent, and a sign)	4	8	28
Double Precision Fixed Point (22 decimal digits and a sign)	12	36	184
Double Precision Floating Point (20 decimal digits, an exponent, and a sign)	16	36	168

2.3 INSTRUCTION OVERLAPPING

To increase the rate of performing instructions without also increasing the amount of equipment in the computer to unreliable and uneconomical proportions, the computer is designed to perform different steps of several instructions in parallel. While one instruction is being executed, an operand for a second instruction is being transferred to or from storage, the operand address for a third instruction is being modified, and a fourth instruction is being obtained from storage. This is illustrated in figure 2-1 which shows a simplified timing diagram for a series of addition instructions. Although any single add instruction actually takes considerably more than four microseconds to perform, a series of these instructions is executed at a rate of one every four microseconds and for practical purposes the total instruction time is considered to be four microseconds. If a transfer of control to a new sequence of instructions occurs, the first instruction of the new sequence requires time to propagate through the several steps before it is executed. Therefore, whenever a transfer of control takes place, eight microseconds is added to the execution time of the instruction that caused the transfer. This time has already been added to the execution times of the transfer of control instructions listed in appendix A.

The technique of overlapping instructions in the computer might upon rare occasions result in a conflict. For example, the results of one instruction might not be available in time if it is required for a beginning step of the next instruction in line for processing. However, such conflicts are resolved automatically by control circuits within the computer.

It should be noted that the timing diagram (figure 2-1) is simplified and serves only to illustrate the technique of overlapping the processing of instructions. Operations not indicated in figure 2-1, such as floating point calculations and error checking, are performed in parallel with the operations shown.

Two of the steps that are overlapped are, obtaining instructions from storage and transferring operands to or from storage. Overlapping these operations requires only that the instructions and operands be assigned to different areas (different units) of the core storage, a requirement that is usually satisfied as a matter of course in any system.

2.4 EXECUTIVE COMMAND OF THE PROCESSOR

The order code of the computer does not include instructions for directly controlling input-output operations. All such operations are handled remotely by the processor under executive command by the computer. Consequently, the computer is free to devote substantially all of its time to performing arithmetic operations. Input, output, and (optionally) editing commands may be summarized by the computer in well-defined pseudo orders (summary orders) having adjustable parameters and issued to the processor singly or in groups via the core storage. The computer program is required only to place the summary orders in the storage, alert the processor to their presence, and check for their completion after enough time has passed for the summary orders to have been executed independently by the processor. Meanwhile the computer can continue with its main program. A list of summary orders that can be executed by one processor program is given in

appendix C. The processor can even be programmed to relieve the computer program of the task of generating and issuing summary orders and, in fact, relieve it of all concern with input-output operations so that computer time can be devoted exclusively to arithmetic and related logical operations. In any case, the computer programmer will be concerned, not with the details of the input-output operations, but with problems of anticipating storage requirements, allocating storage, and directing on a wholesale basis the flow of data to and from the storage.

2.5 MULTIPURPOSE FAST REGISTERS

Any computer instruction that contains an operand address may be tagged with an address of any one of a number of index registers. Before such an instruction is executed, its operand address will be modified automatically by the addition of a constant contained within the specified index register. Arithmetic instructions also contain the address of one of a number of accumulator registers which are used to store an operand involved in an instruction, or the result of an instruction.

The index and accumulator registers are multipurpose fast registers logically interposed betweed the main core storage and the arithmetic system of the computer. They may be addressed and used interchangeably as index registers, as accumulator registers, or in the same way as a standard core storage location. The registers are composed of fast-switching tape-wound cores having a read-regenerate or clear-write cycle of one microsecond. By functioning as fast-access storage for operands and results in either arithmetic operations or address indexing operations, they not only decrease the number of references to the slower main core storage but improve the flexible control of the arithmetic processes. Up to 99 of these multipurpose registers may be included in each computer. All other registers within the computer are non-addressable and ordinarily do not concern the programmer. These include the various control registers and registers within the arithmetic system for storing factors that are used during the actual execution of an arithmetic instruction.

Iterative address modification techniques have been applied in the past only by sacrificing considerable computing time. Often such techniques were used merely as a convenient way of exchanging computing time for storage space in fitting a particular problem to the computer. The fast registers in the LARC computer not only enable the address modification operations to be completely overlapped with the execution time of the instruction but also enable extremely flexible control to be exercised over the iterative processes. An instruction may be tagged to refer to any one of the fast registers for address modification or tagged so that it is not modified. Since a fast register that is used as an index register may also be used in the role of an accumulator register, its contents are subject to all of the arithmetic, test, and other instructions in the computer repertoire. In addition to this, special index instructions (appendix A) are provided which are, in effect, small subroutines for controlling the entry, reentry, and exit from the program loop using control information stored within the index register. A description of the format of the data stored in a fast register when it is used as an index register is given in appendix A.

In one sense, the computer is a single address computer, since each instruction contains only one main storage address. However, any one of a number of fast registers may be addressed and used by an instruction either as an accumulator

register or in the same way as a standard storage location. Furthermore, arithmetic instructions are available which take one operand from one fast register, another operand from the main storage (or another fast register) and place a result in an adjacent fast register. In practical use these features enable the computer to be used much as a three-address computer. Quantities can be accumulated in several fast registers and combined without recourse to separate instructions for returning intermediate partial results to the main storage. To use a very general example, the formation of quantities of the form ab + cd + ef can be effected with a minimum number of instructions.

2.6 TRACING MODE

Any computer instruction may be tagged with any one of nine tracing mode digits. Just before an instruction is executed, the tracing mode digit is detected. If the computer is operating in the designated tracing mode an automatic transfer of control will be effected to a routine associated with the tracing mode. At the completion of the routine, control is transferred back to the main program.

Instructions are available to direct the computer to enter or leave any one of the nine tracing modes by setting a tracing mode flip-flop. For example, when a set tracing mode flip-flop seven instruction is executed, all succeeding instructions tagged with a tracing mode seven digit force a transfer of control to an associated routine. Either the computer program, or the operator with a manual intervention routine, may at any time instruct the computer to enter or leave any of the nine tracing modes. The computer can therefore be directed to automatically trace its programs in several different modes or mode combinations for the purpose of debugging or monitoring a program or for the purpose of injecting side routines at various points in the program. A tracing mode routine can be designed to perform any number of functions. For example, it may be designed to prepare the contents of certain registers for print-out. Except for the instructions that are actually traced, no time is lost since the instructions to be traced are detected automatically by the computer circuits rather than by the program itself. No change has to be made in a program coded for a production run to enable tracing under a number of different conditions.

2.7 SENSE FLIP-FLOPS

The computer instruction code contains instructions for setting or resetting any of ten sense flip-flops together with conditional transfer of control instructions that are dependent on the state of the sense flip-flops. The sense flip-flops have no predetermined function. Essentially, they are general-purpose single-bit storage units that the programmer may use in numerous ways. Like the tracing-mode flip-flops, the sense flip-flops may be set directly by the main program or through manual intervention by the operator. The sense flip-flops enable the program to change the paths it will follow through its own routines or the operator to modify the paths the program will follow. For example, a general program could be directed, by the operator or by the program itself, to perform a specific set of operations on the data for a particular run.

2.8 ERROR CHECKING

In many data-processing installations considerable computing time is wasted in detecting and correcting errors and in handling such program contingencies as overflow. As the speed of data processing systems increase, computing time

becomes more valuable. While a human ponders for a few seconds or manipulates switches on a console, the computer can perform operations numbered in the millions. To prevent the loss of valuable computer time, an error occurring in the LARC system is detected automatically and, whenever possible, corrected without human intervention. If automatic correction proves impossible, the general source of the error is located and isolated from the system, and the exact cause of the error is pinpointed and corrected off line, thereby releasing the system for further computation.

The computer contains built-in checking circuits, designed to detect all single-bit errors. The checking circuits are designed not only to detect an error but also to give an immediate indication of the specific area of the computer in which the error occurred. On the order of 20% of the circuitry in the LARC system is devoted to redundant circuits and associated checking circuits. It is estimated that these circuits double the utility of the system by quickly locating faults and by eliminating the need for programmed checks.

An error may be one of several types. It may be an error due to the complete failure of a component such as a transistor or resistor, in which case the faulty component would have to be located and replaced. It may be an error caused by a transient or intermittent fault which may occur only once or at widely separated intervals. In this case, it would be convenient to repeat the operation that caused the error and continue the program without further interruption after obtaining an indication that the error has occurred and where it occurred. Then again, an error may be caused by an error in the program, for example, by an attempt to add a non-numeric combination or to decode a non-existant address or instruction. In this case it would be convenient for the programmer to obtain information which would aid him in detecting the error and correcting the program.

Should an error be detected in the computer, one or more error flip-flops are set to indicate the type of error and which, when set, cause an immediate and automatic transfer of control of the computer to an error routine designed to handle the situation. Instructions are available for use in the error routines to test the various error flip-flops in order to localize the source of the error. Ordinarily the error routines will not change from problem to problem run on the computer. They are, however, subject to change and improvement by the programmer. Although the error routines may vary considerably in complexity, a typical routine may take the following general form.

When an error is detected, control is transferred to an instruction in a specific storage location which is the beginning of an error routine. The routine examines the error flip-flops to determine the type of error committed and then initiates a printout which would aid the maintenance engineer or programmer in analyzing and correcting the error. The printout might contain the following information:

(1) The type of error (adder, index register, etc.).
(2) The digit position at which the error occurred.
(3) The time at which the error occurred.
(4) The instruction that caused the error.
(5) The storage address of the instruction that caused the error.
(6) The contents of the accumulator register or registers involved, if any.
(7) The contents of the index (B) register involved, if any.
(8) The operand involved.

If possible, the specific instruction that caused the error is then repeated. If the instruction is a type that cannot be repeated, the routine transfers control to a rerun point in the program. In either case, if the error is not repeated, indicating that it is transient or intermittent, the computer continues with the main program without any human intervention having occurred. However, the error is not neglected, since the printout constitutes a running log of all intermittent errors that have occurred, together with information as to their source and frequency. The maintenance engineer may analyze this information to determine what corrective action, if any, is necessary during the next convenient maintenance period.

If an error continues to repeat, indicating that a permanent fault or programming error has occurred, the computer stops. When the computer stops, the maintenance engineer or programmer is usually provided with enough information by means of the typeout or indicators on the engineers console to isolate sufficiently the source of the error. At this point, the maintenance engineer might remove a group of printed circuit packages and replace them with pre-tested equivalents. However, he may find it quicker and more convenient to call in a diagnostic routine from the drums or tapes to further isolate the error to one (or a very few) printed circuit package and replace it.

More elaborate error routines could be designed which would attempt to isolate the error source still further and perhaps print out the designations of the specific printed circuit packages that are to be replaced. However, such routines might not be justified by the frequency of occurrence of the errors.

2.9 CONTINGENCY CHECKING

The computer contains checking circuits for detecting, not only machine errors, but overflow conditions within the arithmetic system and certain conditions reflecting mistakes in programming. These built-in circuits continuously and automatically check for the following conditions:

(1) Floating zero result-occurs on floating point add and subtract instructions when an arithmetic subtraction of two numbers with equal exponents produces a zero answer.

(2) Non-normalized divisor-occurs on floating point division instructions when the divisor has a zero in the most significant digit position (the digit position adjacent to the exponent).

(3) Exponent overflow-occurs on floating point add, subtract, multiply, and divide instructions when the addition, subtraction, multiplication, or division of two floating point numbers results in an exponent greater than 99.

(4) Exponent underflow-occurs on floating point add, subtract, multiply, and divide instructions when the addition, subtraction, multiplication, or division of two floating point numbers results in an exponent less than 00; or on fixed-to-floating point or floating-to-fixed point conversion instructions which would cause a loss of significant digits.

(5) Fixed decimal overflow-occurs on single and double left shift instructions when at least one non-zero digit is shifted out of the register causing a loss of significant digits; or on fixed point add, subtract, multiply, and divide instructions when the result is greater than or equal to one in absolute value.

(6) Program error in sign-occurs on all add, subtract, multiply, divide, negative transfer, and comparison instructions when the rules for the proper character in the sign position are not obeyed.

(7) Non-normalized mistake-occurs on floating point add or subtract instructions when an arithmetic subtraction involving all zero or non-normalized numbers produces the wrong answer.

If one of these conditions arises, it is handled in much the same way as an error condition. The computer automatically transfers control to a contingency routine specifically designed to handle the condition. The programmer decides beforehand exactly what should be done if a particular condition is detected and designs the contingency routine for that condition accordingly. He may, for example, call for a printout to indicate that the condition has occurred but otherwise ignore the condition and return control to the main program; he may decide that the condition be corrected, if possible; or he may decide that the program is at fault and switch to a new program. In any event, the system need not be interrupted while human decisions are made or corrective action taken, and time-wasting programmed checks need not be made to detect contingencies.

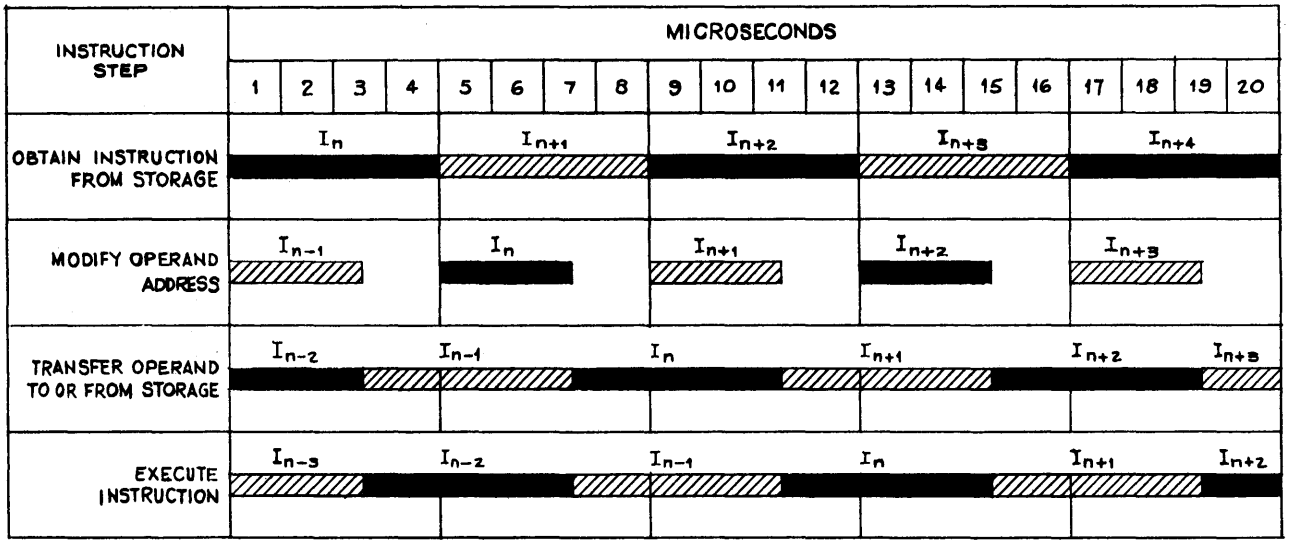


Figure 2-1. Simplified Timing Diagram of a Sequence of Four-Microsecond Computer Instructions

SECTION III

PROCESSOR

3.1 FUNCTION

The processor is a stored-program computer with many general-purpose characteristics. Its primary role in the LARC system is the coordinated and flexible control of concurrent input-output operations under summary command of the computer program.

The processor is programmed to pick up summary orders issued by the computer, acknowledge their issuance, interpret them, supervise their execution, and inform the computer of their completion. All this is controlled by a fluid loop program that need not change for every program run on the computer. A flexible processor program for controlling input and output was developed along with the LARC system much as the equipment was developed and tested. In fact, the program, together with the general purpose computing abilities of the processor, is an alternative to using a multitude of costly and inflexible built-in control equipment to perform a similar function.

An important advantage to the programmed-control approach is that the programmer is given the ability to modify input-output control. As a programmer gains experience with the system he may take advantage of new programming techniques and his own ingenuity to devise control programs for the processor which will without doubt improve the performance and even change the performance characteristics of the processor probably in ways not even contemplated by the original designers.

The processor has enough speed and flexibility to handle the complete complement of input-output devices, operating many in parallel, and to service both computers in the expanded system. Its general purpose characteristics enable the system to be expanded with ease and molded to a customers requirements without conflicting with previously designed computer programs.

When the processor is not being used to the limit of its ability in controlling input-output, as will often be the case, particularly in a one computer system, it can relieve the computer of additional tasks, such as the editing of output data and, if it still has extra time, it can perform a sorting, merging, compiling, or other side routine concurrently with and entirely unrelated to a program being run on the computer. The work load can, in fact, be shared between the processor and the computer so that each does the type of work that it is most suited to do by virtue of its design.

Logically, the processor separates into three major divisions representing different levels of control. The three major divisions are:

(1) The central processor.
(2) The synchronizers.
(3) The dispatcher.

3.2 CENTRAL PROCESSOR

The central processor is the general purpose computing section of the processor. Compared to the computer, it has much less elaborate facilities for performing arithmetic operations. The arithmetic system of the central processor consists of a serial adder-comparator and two connecting 12-digit shift registers for the temporary storage of operands. When an add instruction is executed, operands are shifted from the two registers into and through the adder a digit at a time. The result is shifted back into one of the registers, which also serves as an accumulator register. The bits and digits of a word are transferred between the registers and the main core storage completely in parallel.

The instruction cycle of the central processor is overlapped only to the extent that the access time for obtaining instructions from storage is overlapped with the execution time. Unlike in the computer, the execution time is not overlapped with the time required to transfer operands or perform address modification. The central processor has no special facilities for address modification. The central processor contains the control circuits normally found in most general-purpose computers together with somewhat more specialized control circuits to address, monitor, and exercise supervisory control over the synchronizers and the input-output devices.

3.2.1 Time Reference

The LARC system is designed to change over to a new problem without interruption. While computations are being performed on one problem, the next problem may be in the process of being loaded into the storage, while results from a previous problem are being printed out, all of which is made possible by the parallel operation of the various units of the system.

To aid in a changeover, a real-time reference is provided in the central processor. The timing reference may be used, among other things, to determine when the time allotted for one program on the computer is exhausted so that an automatic changeover may be made to a new program. The timing source is a 60 cycle clock which continually alerts the processor program to keep a running count of the time of day. The computer can order the processor to keep a check on the running time of a program and after a specified period of time has passed direct the computer to a routine for affecting a changeover.

An interesting use of the real-time clock is to time logical operations within the processor. The processor program uses the clock reference to time certain extra long logical operations, such as tape reversal operations on a tape unit or information displays on the operator console, thereby eliminating the need for costly fixed delay elements. Such a timing reference may be used for any number of other purposes, for example, to record the time at which errors or other events occur during the course of running programs. It is particularly useful in scheduling problems on the system and in solving problems in real time.

3.2.2 Instructions

The central processor has an instruction code separate and distinct from that of the computer. It contains a complement of general-purpose instructions (appendix D), such as add, subtract, shift, and transfer instructions, that are used in carrying out its primary editing, interpreting and supervisory functions, but which may also be used in executing a side program. The central processor does not however have instructions for multiplying and dividing since these instructions are not required in the editing and control routines which carry out its primary functions. If multiplication or division is required in a processor program, it could be done by means of a subroutine. More likely it would be done by the processor temporarily interrupting the computer to do it. The central processor can set an intervention flip-flop in either computer which will force the computer to transfer control to a routine associated with the flip-flop. If required, the transfer can be programmed so the computer will return automatically after completing the routine to the point in its own program at which it was interrupted.

The central processor, although it performs arithmetic instructions at a slower rate than the computer, can perform in 16 microseconds an addition instruction which takes an operand from storage, adds it to an operand in the accumulator register, and returns the result to storage.

The majority of the central processor instructions are used to communicate with or control the error circuits, the input-output devices, the synchronizers, and the dispatcher. Most of these instructions are one of the three general types listed below. Many can be addressed to any one of several synchronizers or other devices.

(1) Set flip-flop instructions that alert a synchronizer or other device to perform a specific function, such as connect or read 100 words.

(2) Transfer instructions that transfer control information between the central processor accumulator register and the dispatcher or a synchronizer. The control information might specify the mode in which a synchronizer should operate or the first address of a storage area to which the dispatcher should transfer data for a particular synchronizer.

(3) Test flip-flop instructions that are actually conditional transfer instructions conditional on the state of the flip-flop tested. These instructions are used by the central processor to monitor and test the condition of a synchronizer or other device, for example, to test the availability of a synchronizer or to test if a synchronizer or other device has completed a previously ordered operation.

These instructions are designed chiefly to provide the central processor with the means by which it can exercise flexible real-time control over several operations being performed in parallel. To be meaningful, a description of these instructions requires some knowledge of the characteristics of the various devices which they are used to control and for that reason they are not listed with the more general-purpose instructions in appendix D.

Input and output transfers for a side program of the central processor can be handled in much the same way as it is handled for the computer, by the same processor input-output control program that governs the performance of summary orders from the computer. The programmer therefore need word with a relatively few arithmetic-type instructions when producing a side routine and need not concern himself with a myriad of detailed input-output orders. Such orders would only be used if the programmer wished to modify or redesign a processor input-output control program.

3.3 SYNCRHONIZERS

The central processor does not have time to control every step of several input-output operation being performed in parallel. Consequently, much of the detailed and specialized work of controlling these operations is performed by the synchronizers. A synchronizer represents a modular grouping of logical circuits for controlling a particular reading or recording process. Its logical form depends to a great extent on the characteristics of the reading or recording device with which it is associated. Physically the synchronizers are contained within the processor cabinet which has the potential for accepting all of the synchronizers of the expanded system listed in Table 1-1.

3.3.1 Function

The synchronizers control the actual reading or recording process, and the translation and the serial flow of information, digit by digit, between a buffer register and an input-output device. In the process of transferring information a synchronizer may perform such functions as synchronizing input information with the internal timing of the system, checking and counting the information for errors, or translating the information in one way or another.

Whereas a single summary order from the computer might call for the transfer of a block of several hundreds or even thousands of words, a single instruction by the central processor alerts the synchronizer to process a smaller block of words which is some fraction of the block specified by the summary order. The synchronizer is designed to process the smaller block automatically without direct intervention by the central processor. While one block of information is being processed, the central processor program may or may not, based upon parameters defined in the summary order from the computer, alert the synchronizer to process to next block of information. The actual instruction that alerts the synchronizer is executed by the central processor in four microseconds whereas the operation initiated in the synchronizer or other device by the instruction may take several milliseconds. Meanwhile, the central processor is free to continue processing other summary orders.

3.3.2 Communication with Central Processor

Whenever a synchronizer or an input-output device completes an operation or action to a point where it is ready to accept further instruction from the central processor program it records this fact by setting a flip-flop. The central processor program tests the flip-flop with a conditional transfer instruction to determine whether a synchronizer or other device requires attention. If it is set when tested, control is transferred to a routine which determines what action to take and instructions the device accordingly.

Rather than waste time testing every flip-flop to find the one that is set, the central processor program short-cuts the testing by first performing a master test on all of the flip-flops and then a series of group tests. The testing is performed in an order of priority so that the flip-flops of the synchronizers that require the most frequent attention are tested first. In general, the frequency with which a synchronizer requires attention is a function of its data transfer rate. Consequently the drum synchronizers which operate at the highest rate are tested first, then the tape synchronizers, etc. A synchronizer is designed to alert the central processor of its need for instruction sufficiently far in advance to give the central processor program time to complete any routine in which it happens to be engaged and satisfy any requests for supervision by higher priority synchronizers. Any routine run on the central processor is required to perform the master test periodically to monitor the real-time operations of the synchronizers.

3.4 DISPATCHER

The dispatcher is a central exchange which controls the transfer of data between the buffer registers of the synchronizers and the main core storage. One to four one-word buffer registers for each synchronizer perform the serial-parallel conversions and store data preparatory to transferring it to the main storage or to an input-output device. Transfers between the buffer registers and the input-output devices are controlled by the synchronizers, and the rate and order of flow is governed primarily by the characteristics of the particular device concerned. Whenever a synchronizer completes the processing and transfer of a word to or from a buffer register, it signals the dispatcher to transfer the word to the main storage (if it is an input synchronizer) or to obtain a new word from storage (if it is an output synchronizer). To accomplish this, means must be available to the dispatcher for keeping track of the storage address of every word transferred. Therefore, a register is provided in the dispatcher for each synchronizer in which is stored the main storage address of the next word to be transferred to or from the buffer register of that synchronizer. The beginning storage address for a particular operation is supplied by the central processor from information derived from the summary order from the computer and the address is modified automatically each time the word it represents is transferred. Whenever an address is read out of an address register and sent to the storage to initiate a transfer, it is also sent to a time-shared address modifier in the dispatcher where it is modified before being written back into the same register at the end of the transfer operation. The modified address is then the storage address of the next word to be transferred for the synchronizer.

Since the synchronizers process information asynchronously with respect to one another, several may require access to the storage at the same time. Priority circuits are therefore contained within the dispatcher to define, on the basis of preassigned priority ratings, the time at which transfers are made for each synchronizer. The priority ratings, in general, reflect the rates at which the various synchronizers must have information transferred to maintain, without interruption, the flow of data required by the input-output device.

3.5 PROCESSOR CONTROL PROGRAM

A list of summary orders that are accepted and executed by one control program designed for the processor are listed in Appendix C. The way in which the program receives and sequences the execution of the summary orders is described briefly below. The processor control program described here is a general-purpose program designed to be applicable with practically any type of program run on the computer. However, it does not by any means represent the most efficient method of controlling the input-output operations for any one type of computer program. Much more efficient processor programs can be compiled which, without receiving summary orders, execute the input-output operations independently, in an order and sequence compatible with the requirements of the computer program being run. Such a processor program relieves the computer of the tasks of constructing and issuing summary orders and, at the same time, relieves the processor of the tasks of receiving, extracting, and interpreting the information summarized in the orders. Direct exchange between the computer and processor ordinarily would be limited to the computer at various points in its program alerting the processor to begin an extensive series of input-output operations in a predetermined sequence and the processor, in turn, alerting the computer to the completion of the series.

3.5.1 Summary Orders

The summary orders listed in Appendix C are not instructions in the sense that they are decoded by a built-in decoder in the processor. Rather, they are in the form of a pseudo or program code interpreted by the processor program to generate actual central processor instructions which control the synchronizers and other devices in carrying out the summary orders. A summary order usually specifies a type of operation to be performed, the drum or input-output device involved in the operation and parameters of the operation. For example, a tape summary order may specify that 250 blockettes of 10 words each be read in a forward direction on tape unit 15 and be transferred to consecutive storage locations beginning with storage location 15,600.

3.5.2 Disclosure of Summary Orders

A computer program communicates with the processor by placing one or more such summary orders in sequential locations in the main storage and alerting the processor program to their presence by setting a disclosure flip-flop. A count of the number of summary orders in the group that call for a transfer of data to or from the main storage is contained in a word filed along with the summary orders. Whenever the processor program completes one of these summary orders, it lowers the count by one so that the computer program can check the count for zero to determine that all required data transfers have been accomplished. As an alternative, the computer program can code the word containing the count of summary orders in such a way that the processor program will be directed to interrupt the computer program and transfer computer control to a specified subroutine as soon as all the required data transfers are completed.

The central processor program, upon testing the disclosure flip-flop, is directed to a perdetermined storage location which contains a disclosure word (the first and last address of the summary orders), placed there by the computer, that defines the area of the storage in which the summary orders are stored. After picking up the information which defines the location of the summary orders, the processor program resets the disclosure flip-flop, thereby enabling the computer to issue another group of orders. Before issuing a new group, the computer program tests the disclosure flip-flop to ensure that the processor program has been alerted to the presence of the last group that was issued.

3.5.3 Execution of Summary Orders

The processor program is designed to accept summary orders and execute them as rapidly as possible, relieving the computer program of all concern about the availability of the equipment needed to complete the summary orders. Once the processor program has been alerted to the presence of the summary orders, it examines and begins to execute each summary order in turn, provided all of the equipment that is needed to process the summary order is available. Delays in processing a summary order occur if:

(1) The central processor is temporarily tied up supervising operations already in progress.

(2) A drum, input-output device, or synchronizer that is needed to execute the summary order is busy performing a previously ordered operation.

(3) The summary order calls for a transfer of data to or from the same area of main storage as a previously accepted summary order which is not yet completed.

Should condition (2) or (3) exist, the processor program files the summary order and attempts to execute the next one.

To ensure their continuity, the processor program always gives priority to the control of operations already in progress. At least every 500 microseconds, the central processor program performs the master test to monitor and answer all requests for instruction by currently operating devices. Only if all requests for instruction by currently operating devices have been satisfied will the program accept for processing newly issued summary orders.

Once accepted, a new summary order is started in execution by the processor program if the necessary equipment is available to process it. Should a needed device (drum, input-output device, or synchronizer) be busy performing another operation, the summary order is stored in a work order file for the device that is holding up execution. Each time a device completes an operation the next summary order, if any, in the work order file for the device is picked up by the program and is started in execution.

The processor program also keeps a conflict control file of all summary orders currently being executed or awaiting execution that call for a transfer of data to or from the main storage. As each new summary order is received, comparisons are made by the program to determine if the order is in conflict in that it refers to the same storage area as an order in the conflict control file. If there is a conflict, then the program delays the execution of the summary order, even though the equipment needed to execute it is available, until any previously accepted summary order with which it is in conflict has been completed. This prevents the possible premature destruction of data in the storage or the transfer of wrong data from the storage as a result of a summary order being executed out of sequence.

3.6 ERROR CHECKING

The processor, like the computer, contains checking circuits throughout that are designed to detect all single bit errors. An error in the central processor is handled in somewhat the same manner as in the computer (paragraph 2.8) in that an error flip-flop is set causing an immediate and automatic transfer of control to an error routine which can test the error flip-flops and take appropriate action.

Should an error be detected in a synchronizer it is handled differently. Two flip-flops are set, one, which may be set by error-checking circuits of a particular type in any synchronizer, indicates the type of error detected; the other, which is set only by the error-checking circuits of the one synchronizer, indicates which synchronizer detect an error. When the error flip-flops are set by the synchronizer, the central processor does not immediately transfer control to an error routine. However, the next time the central processor performs an instruction which alerts the synchronizer to process another block of data or to end an operation, the instruction will test the error flip-flop of the synchronizer to determine if an error was committed during the processing of the last block of data. Should the flip-flop be set when tested, the central processor will transfer control to an error routine.

The central processor has error test instructions which can be used in the error routine to determine the type and location of the failure. It therefore can keep a running log of detected failures and, if possible, initiate a short rerun procedure. A rerun procedure in the processor may take the form of rereading a block of data from a magnetic tape. Instructions are provided the processor for setting the read-amplifier gain control of the tape units to normal, high, or low gain. A block of data that cannot be read at normal gain often can be salvaged, without human intervention, by rereading it at a different gain setting.

The checking circuits within the processor are designed to indicate whether an error originated within the logical circuits of the processor or within an external device (storage or input-output device). Often it is possible to exchange a good external device for a faulty one and continue with computation. The tape units, drums, and core storage units are all connected into the system through plugboards which simplify the substitution of one identical unit for another. The external devices are provided with test controls and other provisions for "off-line" testing and troubleshooting without tying up the complete system.

SECTION IV

MAGNETIC CORE STORAGE

4.1 FUNCTION

The magnetic ferrite-core storage of the LARC system is the main storage for both computers and the processor and the common link through which they exchange information. It is also used as a flexible buffer storage for blocks of data being transferred between the drums and input-output devices. Any area of the storage not being used for other purposes may be used as temporary buffer storage.

4.2 MODULAR CONSTRUCTION

The core storage is divided into modular units each of which has a capacity of 2500 words of 12 decimal digits. Four storage units are contained in a cabinet and the storage units may be added to a system in units of four up to a maximum of 39 units, the equivalent of 97,500 words. Each cabinet is self-contained. It has its own power supply, clock-pulse generator, and heat exchangers. Because of a logical limitation on the number of storage addresses available for assignment, one cabinet in a completely expanded storage system of ten cabinets would contain only three 2500-word units. Although the core storage may be expanded to a capacity of 97,500 words, with the drum storage to back it up, less would ordinarily be required for most systems. For example, a system containing one computer that is being produced for the Livermore Atomic Research Center contains only eight core storage units with a total capacity of 20,000 words.

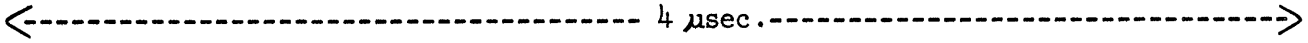
Each storage unit contains the switching, timing, and amplifying circuits required for independent operation. The division of the storage into units capable of independent operation enables simultaneous references to be made to the storage both by the computer, for obtaining instructions and for transferring operands, and by the processor, for instructions or data involved in carrying out its program and for transferring data to or from the input-output. It also enables off-line maintenance to be performed on a single unit while the others are operating. A plugboard is provided to functionally interchange the units, that is, change the address assignments of the units.

4.3 TIMING

With the circuits used in the core storage a complete clear-write or read-regenerate cycle would normally take on the order of eight microseconds to complete. However, the allowable transfers to or from a single storage unit are increased to a rate of one word every four microseconds by designing the storage units so that the operations of selecting the proper cores and reading or clearing data from them are overlapped and performed in parallel with the operations of selecting the cores and writing or regenerating the data. As a result, the storage cycle is, for all practical purposes, four microseconds.

The allowable transfer rate to or from the storage is further increased by the fact that the storage units can operate independently and in parallel. To keep interconnections within reasonable limits, the core storage units are connected to the computers and the processor by a transfer bus which is time shared to serve as the data transfer path to and from the storage for

both the computers and the processor. The bus is time shared on the basis of a repetitive 4-microsecond time cycle which is broken down into eight one-half microsecond time intervals (time slots) during any one of which one word of data can be transmitted to the storage in parallel. The time slots are apportioned to the various connecting areas of the computers and the processor in the following order.



P CP	C1 I	C2 O	P D	Not Used	C2 I	C1 O	P D

The abbreviations and their meanings are:

P = processor time slot

CP = central processor; indicates the time slot on which the central processor section of the processor transfers operands or instructions that are used in its program.

D = dispatcher; indicates the time slots on which the dispatcher section of the processor transfers input-output or drum data for a synchronizer.

C1 = computer number 1

C2 = computer number 2

I = Instruction; indicates the time slot on which a computer obtains an instruction.

O = Operand; indicates the time slot on which a computer transfers operands.

Each storage unit is independently timed so that it can begin its operating cycle on any one of the time slots. Consequently, disregarding the one time slot that is not used, data may be transferred to or from different storage units at a maximumn rate of one word every one-half microsecond. Logical operations within the computer and the processor are timed so that the storage transfers required in the performance of the operations coincide with the time slot allotted to the particular section of the computer or processor performing the operation. All instructions have been designed to require some multiple of four microseconds for their execution, so that their execution times are compatible with (in phase with) the four microsecond rate at which the instructions can be obtained from the storage. All data transferred over the bus system by either the computers or the processor are checked for single bit errors per decimal digit by a single set of time-shared error checkers located in the processor. Error signals from the checkers are distributed to the sections of the system that originate the transfers over a separate set of time-shared buses.

4.4 INTERLOCKS

The LARC system ordinarily is programmed to avoid simultaneous reference to the same storage unit by two or more sections of the system (the computers, the central processor, or the dispatcher). However, occasions do arise in which such a situation cannot be avoided. Each storage unit therefore is provided with interlocks to prevent conflicts and to establish priorities. A storage unit is unavailable for four microseconds from the time it is addressed by a calling unit. During this time any other calling unit is prevented from gaining access to that storage unit. Continual reference, once every four microseconds, to a storage unit by a computer or the central processor could, if a system of priorities did not exist, lock out the dispatcher indefinitely from access to the same storage unit. Because the dispatcher, to maintain a continual flow of data for the input-output operations, must transfer data for the synchronizers within a definite period of time, each storage unit is provided with a priority interlock which ensures access by the dispatcher within four microseconds after it addresses the storage unit. Should the dispatcher address a storage unit that is busy, the interlock prevents the computers or the central processor from gaining access to that unit again until the dispatcher has successfully completed its storage reference. Because the central processor must monitor within a definite period of time the real-time operations of the synchronizers, it has the same type of storage priority over the computers.

The processor instruction repertoire contains write-interlock instructions to prevent either computer from writing data into any designated storage unit. It also contains an instruction for removing the write interlock from any designated storage unit. These instructions are provided to prevent the accidental destruction by the computer of data in the storage, for example, to prevent one computer from destroying the processor's or the other computer's program.

SECTON V

MAGNETIC DRUM STORAGE

5.1 GENERAL

The main core storage is backed up by a magnetic drum storage which is intermediate in speed, cost, and capacity between the core storage and the magnetic tape storage. The drums serve as a repository for information which is not required immediately for a current series of computations or for a current output operation but which will be or is likely to be required for problems currently being run or scheduled to be run next. Information stored on the drums might include input data and instructions for current and future problems, output data for current and previous problems, intermediate results for a current problem, and service routines.

Up to 24 magnetic drums may be included in a system. Each drum can store 250,000 words of 12 decimal digits. Up to three read-synchronizer and two write-synchronizer units can be added to the processor enabling it to control concurrently as many as three reading and two writing operations on five different drums concurrently with the input-output operations. The processor program can connect any drum to any synchronizer. Two drums operating alternately can, with a single synchronizer, transfer data at a continuous rate of 330,000 decimal digits per second.

The drum units are self-contained modules with individual test controls for off-line troubleshooting and maintenance. They may be functionally interchanged by means of a plugboard.

5.2 FLOATING HEAD ASSEMBLY

Physically the drum consists of a brass tube, 27.6 inches long and 24.2 inches in diameter, electroplated with a nickle-cobalt alloy. The drum is rotated at 880 rpm to attain a surface velocity of 1120 inches per second. The complete 250,000-word drum is serviced by a single six-channel read-write head assembly held in a gimbal mounting and floated on a thin film of air that is dragged along by the surface friction of the rotating drum. The head assembly is maintained by the air film at a fraction of a mil from the surface of the drum thereby enabling the heads to record and read at a pulse density of 450 pulses per inch. Because the close spacing can be maintained despite thermal, dimensional, and surface variations on the drum, it is possible to use a relatively large, high-capacity drum and a single head assembly which is moved across the length of the drum to service the complete drum. If a fixed head were used, the tolerances involved could be relaxed to practical proportions only by decreasing the size or pulse density, and therefore the capacity, of the drum and by providing multiple, costly head assemblies (with additional amplifying and switching elements) mounted along the entire length of the drum. To protect it against the effects of dirt and moisture, the drum, together with the head assembly and the mechanism that positions the head assembly, is housed in an air-tight dust-free enclosure.

5.3 ORGANIZATION OF DATA

Each drum has 100 circumferential information bands. Each band can store 2500 computer words of 12 decimal digits, a capacity equal to that of one core storage unit. The bands are divided into 25 sectors of 100 words each, making it possible to begin reading or recording at any one of 25 access points around a band. One sector is the smallest unit that can be read or recorded during any one reference to the drum. A band is formed of six tracks. The four information bits and parity check bit of a decimal digit are recorded in parallel on five of the tracks. Words and digits of a word are stored serially. The sixth track of a band is used to store serially a band number and an address for each sector. A self-sprocketing, phase-modulation system of recording is used in which synchronizing sprocket pulses are derived from the recorded information pulses themselves.

Access to a particular band is gained by moving the single six-channel head assembly back or forth along the length of the drum. The six individual read-write heads are spaced in the head assembly at twice the track spacing on the drum and the six tracks of one band are always interspaced with the tracks of another. Two bands are interlaced by recording one band of an interlaced pair with the head assembly in one position and recording the other band of the pair with the head assembly shifted a distance equal to the track spacing. This is illustrated in figure 5-1 which shows a graphical representation of a small section of the drum and relative head assembly positions. Interlacing the bands in this manner enables a high track density of 28.5 tracks to the inch to be achieved without difficulties attendent to designing a head assembly with extremely close spacing between the individual heads. Three distinct operations are involved in positioning the head assembly: stepping the head assembly from one band of an interlaced pair to the equivalent band of an adjacent pair (figure 5-1), shifting the head assembly from one band to the band with which it is interlaced, and reversing the direction of stepping.

The 100 bands of a drum are numbered in the following order:
00,99 01,98 02,97. . . . . 47,52 48,51 49,50
where 00 and 99 are the numbers of the first pair of interlaced bands, 01 and 98 are the numbers the next, etc. To take full advantage of this system, data should normally be organized in a systematic manner so that the processing of data begins at band 00 and proceeds through band 01, 02, 03, etc. During each read or write operation, a complete band of data is processed after which the head assembly is stepped, in sequence, to the next band. After the data on band 49 is processed, the head assembly is shifted to band 50, the stepping mechanism is reversed, and the head assembly is stepped in the opposite direction through bands 51, 52, 53, etc. After a complete pass through the drum the head assembly is back to the starting point, thereby eliminating the equivalent of rewind time. Data can be organized on the drum in a less systematic manner and less than a full band can be read during each reference, although by so doing the time required to obtain data would ordinarily be increased. The computer program can order the processor program to position the head assembly over any band and process any number of sectors from 1 to 25.

5.4 ACCESS TIME

Stepping the head assembly from one band of an interlaced pair to the equivalent band of the next pair requires 70 milliseconds except when the head assembly is continously stepped without reading or writing. In this case only 50 milliseconds is required for each step except the last which requires 70 milliseconds. To reverse the direction of stepping requires 10 milliseconds. Shifting the head assembly from one band to the band with which it is interlaced requires 50 milliseconds. The shifting operation can be performed in parallel with a reversal operation, however.

Positioning of the head assembly is controlled by the processor program which keeps a running account of the current position of the head assembly of each of the drums and moves the head assembly to any position ordered by the computer. The circuits in the processor that control the head positioning are independent of the drum synchronizers. The head assembly on a drum can be positioned in parallel with the head assembly on any other drum or in parallel with a read or write operation being performed on any other drum.

A single summary order from the computer can specify the transfer to or from sequential main storage locations of from 1 to 25 sectors of data on a band. If a complete band of 2500 words is to be transferred, reading or writing can start as soon as the beginning of the next readable sector is reached after the drum is connected to the synchronizer. If only one specific sector on a band is desired, up to a full drum revolution of latency time (68 milliseconds) may be required to gain access to it.

The head assembly on any drum can be moved to the next band in sequence during the time required to read or write a full band of data. By organizing the data in such a way that the reading and writing of bands alternate between two drums, head movements can be executed in parallel with the reading and writing, that is, while a band of data is being processed on one drum the head assembly on the other drum can be moved to the next band in the sequence. With this technique, data can be transferred continuously at a rate of 2500 words every 90 milliseconds, the equivalent of 330,000 decimal digits per second. The 90 milliseconds required to transfer 2500 words includes:

(1) The time required to pick up and interpret the summary order from the computer.

(2) The time required to connect the drum to the synchronizer (assuming that the required drum and synchronizer are available).

(3) The time required to gain access to the first sector that is processed.

(4) The time required to transfer the complete band of data to the main storage and notify the computer of the completion of the transfer.

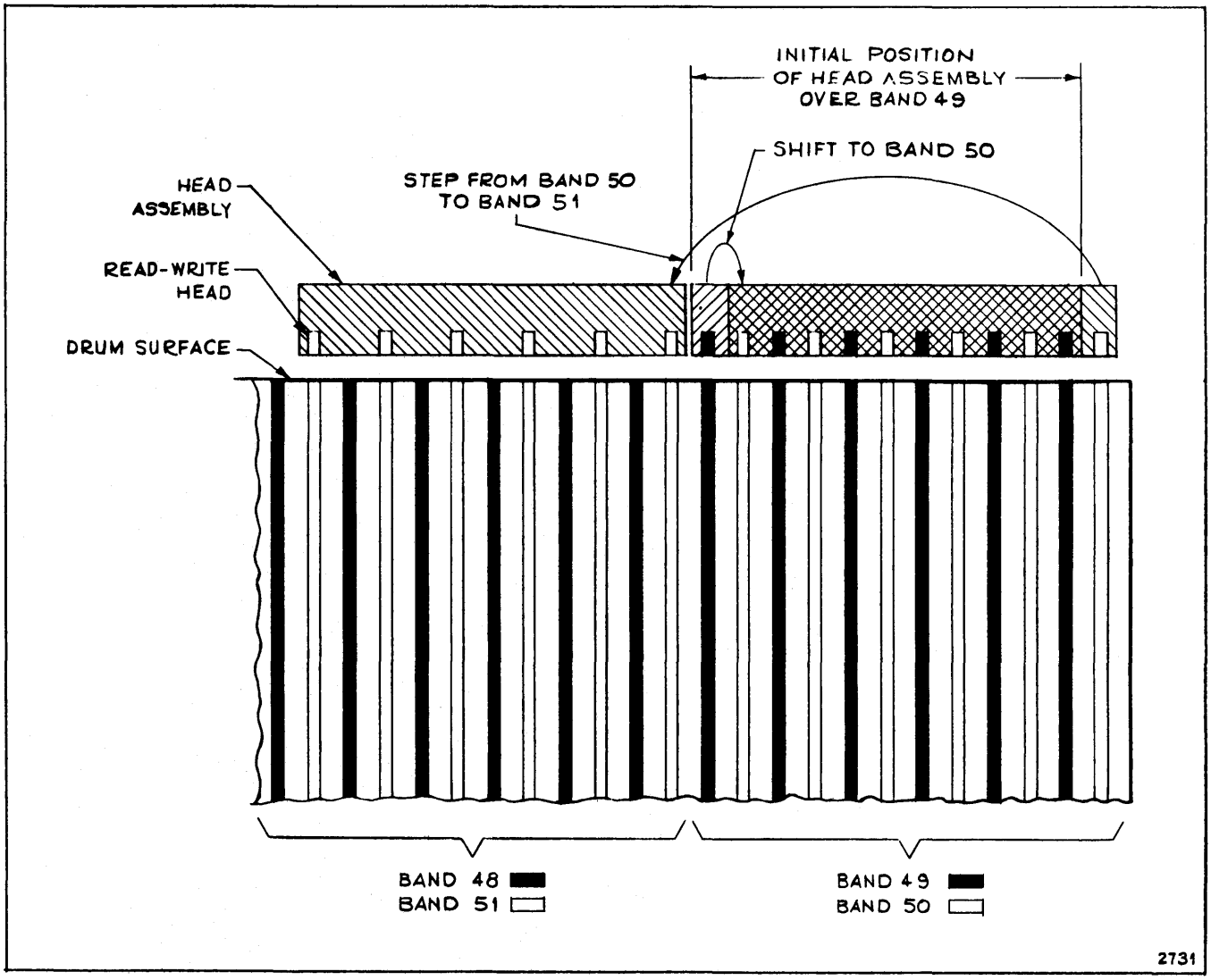


Figure 5-1. Section of Drum Showing Relative Head Assembly Positions.

SECTION VI

INPUT-OUTPUT

6.1 MAGNETIC TAPE UNIT

High Speed input and output is provided in the LARC system by Uniservo II magnetic tape units. The tape units, although they can be considered and used as a third order storage, are used primarily to introduce data into the system and to record output, either for long-term storage or for off-line conversion on an auxiliary device such as a high speed printer. Any magnetic tape prepared by the Univac I or Univac II central computer or Univac auxiliary equipment can be read by LARC system magnetic tape units and, conversely, LARC system tape units can prepare tapes for use by the Univac I or Univac II central computer or Univac auxiliary equipment. Consequently, a fully compatible complement of off-line auxiliary equipment can be used with the LARC system, including the Univac High-Speed Printer, Unityper I, Unityper II, Tape Verifier, Card-to-Tape Converter, and Tape-to-Card Converter.

The LARC system can accommodate any number of Uniservo II tape units up to 40. As many as four modular tape control units (synchronizers) can be included in the processor, each of which can control a reading or recording operation on any one of 10 tape units that can be connected to it. Plugboards make it possible to substitute any tape unit for any other controlled by the same synchronizer. The four tape read-write synchronizers can perform in parallel with one another and in parallel with the drum synchronizers or other input-output synchronizers in the system. While a reading or recording operation is being performed on one tape unit, the tapes on any one or all of the others can be rewinding.

A rewind checker can be provided as part of the processor. The rewind checker may be connected by the processor to any tape unit via its synchronizer. It is used to perform a parity check and count of the characters recorded on a reel of tape as the tape is being rewound to ensure that the recording was executed without error. Although the rewind checker uses the read circuits of the tape synchronizer to which it is connected, the same synchronizer is free to control a concurrent write operation on another tape unit.

The tape units read or record on either plastic or metal tapes at a tape speed of 100 inches per second. Data can be recorded at a density of 200 numeric or alphanumeric characters to the inch for use on the Larc or Univac II systems, or at a density of 100 characters per inch for use on the Univac I system or on Univac off-line auxiliary devices. Tapes having a wider range of pulse densities can be read, including tapes recorded by the Unityper I at 20 pulses per inch. The length of a block of data on the tapes can be any multiple of ten 12-digit words. The tape synchronizers perform a parity check and count of the characters that are read or recorded in each 10-word group of a block. The processor program checks the number of 10-word groups in each block that are read or recorded. The tapes can be recorded in a forward direction and read in either a forward or backward direction. The data read from the tapes can be transferred to the main core storage or merely checked by the processor program in order to pass a number of blocks of data and gain access to a particular block.

All data recorded on the tapes is in Univac excess-three 7-bit code. Input and output data are translated by built-in translators in the tape synchronizers. The processor program can instruct the tape synchronizer to translate the Univac 7-bit code read from the tape into either a 1-digit numeric or 2-digit alphanumeric code used within the Larc system (appendix B). When output is being recorded on tape, the synchronizer can be instructed to translate from either the 1-digit numeric or 2-digit alphanumeric code to the Univac 7-bit code.

A mechanical interlock is provided on each tape unit to prevent writing on a designated reel of tape. Also provided on each tape unit is a rewind interlock that can be set by the processor program to prevent reading or writing on a rewound tape until the operator releases the tape unit from the interlock.

A photocell on each tape unit detects unusable sections of tape that are indicated by holes punched at 2 1/2-inch intervals through the center of the tape. When an unusable section is detected, the reading or writing process is interrupted until it has passed the read-write heads. The tapes may be easily spliced and the spliced joint invalidated for reading or recording by punching spaced holes.

Access time to the nearest block of data on the tape is 15 milliseconds. This includes the time required to connect the read-write heads and accelerate the tape. An additional 0.6 second is required if the tape direction is reversed. When a tape is in a rewound state an additional 1.2 seconds is required to reach the first block of data on the tape.

6.2 ELECTRONIC PAGE PRINTER

For direct large-volume output of data in tabular or graphical form, an electronic page printer can be provided with the LARC system. Output data may be represented by the printer in the form of a curve plot, a grid pattern, alphanumeric characters, or a combination of all three, that is, a plotted curve with call-outs, titles, scales, grid patterns, etc. The output is displayed on the face of a cathode ray tube and is recorded by a high-speed 35-mm camera controlled by the processor program. For occasional monitoring of the output, a self-developing camera is provided.

The printer operates at an average character rate per frame of approximately 15,000 characters per second. When it is used for graphing, the printer operates at average rates per frame of approximately 2000 points per second or 1000 grid lines per second. Such output rates begin to match internal computer speeds and make it possible to produce a sufficient volume of timely and easily interpreted output data for program debugging and efficient engineering and mathematical analysis.

Two identical fast-interchange printers are provided with the system. Each has a 35-mm and a self-developing camera. Both printers are controlled from a single synchronizer in the processor. Each printer is provided with an internal test program generator which may be used to simulate processor instruction for purposes of off-line maintenance and adjustment. Should one printer become inoperative or its camera run out of film, all of the output can be recorded on the other.

The format for printing numeric or alphanumeric characters contains a maximum of 65 lines of 125 character positions each. Any one of 64 characters or symbols can be recorded (appendix B). The printer synchronizer can operate in either a numeric or alphanumeric mode. In the numeric mode, a single digit in LARC code is used to select a numeric character or one of five special symbols. In the alphanumeric mode, 2 digits in LARC code are used to select any one of 64 alphanumeric characters or symbols. The printer synchronizer can operate in an unedited mode in which case all of the characters are recorded in a standard format, or in an edited mode, in which case the format is determined by digits, within the output data itself, which do not print but instead perform non-printing functions such as inserting a space.

Any one of the 64 characters or symbols can be selected by the processor program for use in plotting a curve. The center of the plotting symbol can be directed to any position in a 1000 by 1000 mesh of discrete locations. The printer synchronizer can operate in either of two plotting modes. In one mode two sets of X and Y coordinates (2 points) are specified in a single word of output data. In the other plotting mode, X and Y coordinates for one point are specified in two consecutive words of output data. The output data can be edited by a special editing routine in the processor or the computer. The synchronizer can operate in two additional modes, one for plotting horizontal grid lines and the other for plotting vertical grid lines. In these two modes, a pair of abscissas or ordinates is specified in each output word for plotting respectively, a vertical or horizontal grid line.

Both the 35-mm camera and the self-developing camera are focused on the face of the cathode ray tube by a "beam-splitter" optical arrangement. The 35-mm camera has no shutter. Its film is advanced automatically, with the tube cut-off, after a frame of data has been recorded. The film can be advanced at a maximum rate of 10 frames per second. The camera accepts a 400-foot film magazine. Both the processor program and the operator are provided with indications of the status of the film.

The shutter for the self-developing camera is controlled by the processor program. The camera can produce either standard paper prints or positive transparencies that are suitable for projection and for easy overlay comparison of graphs. The operator can advance the film and develop a paper print in approximately 1 minute, or a transparency in approximately 2 minutes.

6.3 LINE PRINTER

Either one or two electro-mechanical printers and printer synchronizers can be connected into the system to produce directly high-quality, multiple-copy records of results. The printer can record either numeric or alphanumeric data in a standard or a completely edited format. The paper can be fed in steps for single-line spacing or fast fed for multiple-line spacing. The paper feed accepts paper from 4 1/2 to 21 inches in width. The printer has the following characteristics:

Printable characters	51
Lines per minute	600
Character positions per line	130
Characters per minute (maximum)	78,000
Characters per inch	10
Lines per inch	6

6.4 CARD READER

An on-line card reader and a card reader synchronizer can be provided as part of the LARC system.

SECTION VII

OPERATOR AND ENGINEER CONTROL CONSOLES

7.1 OPERATOR CONTROL CONSOLE

In a LARC system containing one computer, a single operator console is used to exercise complete operational control over both the computer and the processor. In a LARC system containing two computers, two identical operator consoles are used, one for each computer. When separate problems are being run on the two computers, the operators at both consoles can communicate with and exercise control over the processor without in any way interfering with one another. Although the controls and monitoring devices on the operator control console are few, they are versatile enough to provide complete operational control over both the computer and processor.

Each operator console has a numeric keyboard to enter data into either a 5-digit or 12-digit general-purpose display register in the computer. The contents of both registers are displayed in decimal form on the operator control console panel. Both the computer and processor have access to the display registers by way of their accumulator registers and may be instructed to display data in the registers or pick up data entered into the registers from the keyboard. To prevent conflicts between the computer and processor in the use of the display registers, all displays are normally handled by the processor program which times the duration of the display using the real time reference.

Operator direction of the computer and processor is exercised chiefly by means of manual intervention buttons on the operator control panel. There are five such buttons on each panel for the computer and five for the processor. Pressing a computer manual intervention button forces the computer to transfer control to a routine associated with the button. If necessary, provisions are made in the routine for reentry into the main program when the routine is completed. The routine may take any form and may, for example, pick up data entered into the display registers by the operator and use it, or dispose of it, in any number of ways. Although only five manual intervention buttons are provided for the computer, the number of routines to which the operator may direct the computer is not necessarily limited to five. For example, the data picked up from a display register may be interpreted by a manual intervention routine in such a way as to direct the computer to any number of subroutines. A subroutine may perform any one of a variety of functions such as fill, empty, switch to another problem, set a sense flip-flop, or set a tracing mode flip-flop. Although the manual intervention buttons are often used in conjunction with a display register entry, they need not be.

Pressing a processor manual intervention button causes a transfer of control in the processor to a routine associated with the button. Instead of forcing an immediate transfer, however, the button alerts the processor to transfer control at the discretion of its program. The actual transfer may not be immediately affected but may be delayed a few milliseconds until the processor has reached a point in its program at which it may enter the routine without interfering with a current input-output operation or an intervention by an operator at another console.

An alphameric typewriter printer is provided on each operator control console. The printer is controlled through a synchronizer in the processor. It is used by the processor program, or by the computer program indirectly by way of summary order to the processor program, to communicate with the operator. A printout might consist of data relative to errors or contingencies (that occur in a program), or programmed instructions to the operator.

A paper tape reader is provided with the printer on each console. The paper tape reader is used primarily during a start-up procedure as a fast means of initially loading the first of the processor program into the main storage. When used for this purpose, an initial load button on the operator control panel is pressed which causes data read from the tape to be transmitted to the main storage via the display registers. After sufficient data have been transferred into storage from the paper tape, the processor program completes the loading from magnetic tape. The paper tape reader may also be used to relieve the operator from typing data into the display registers from the keyboard while a problem is being run or to load data into the storage for the computer if the processor is not operating. The printer is also provided with a paper tape punch which may be used to record the numeric output of the printer or may be used by the operator to prepare punched paper tapes for the reader.

Aside from the manual intervention buttons, the initial load button, and the keyboard controls, the only other controls on the operator console are five illuminated push buttons -- a START, STOP, CONTINUOUS, ONE INSTRUCTION and MULTIVIBRATE button. Pressing the multivibrate button causes the computer to perform instructions at a frequency determined by a dial setting on the engineer control console. The one instruction button causes the computer to perform one instruction then stop. When the continuous button is pressed, the computer performs the instructions at normal running frequency.

The only indicators on the operator panel, aside from those associated with the display registers, are:

(1) A 5-digit decimal display of the contents of the control counter in the computer.

(2) Nine tracing mode indicators to indicate the tracing modes the computer is operating in (which tracing mode flip-flop is set).

(3) Ten sense flip-flop indicators to indicate which computer sense flip-flops are set.

(4) Seven interlock indicators which are general indicators of certain conditions requiring operator attention with respect to the drums and each type of input-output device (magnetic tapes, line printers, electronic page printers, card reader, console printers, and console keyboards). More specific data concerning a condition would be indicated at the engineer control console or at the device itself.

7.2 ENGINEER CONTROL CONSOLE

The engineer control console contains an engineer control panel for the processor and a separate panel for each computer in the system. All of the controls and indicators on the operator control console which includes an operator control panel, a numeric keyboard, and an alphanumeric printer with an associated paper tape punch and reader are duplicated on the engineer control console.

In a system containing two computers, either one or two sets of operator controls and indicators can be provided on the engineer control console. Should one set be provided, it can be manually connected to either computer. A single console printer synchronizer is provided in the processor which can be connected by the processor program to any one of the four console printers in the system. Normally, error analysis information and the like is routed to the printer at the engineer console and data on contingencies, instructions to the operator, etc. are routed to the printer at the operator console. Some, but not all, of the controls and indicators provided on the computer and processor engineering panels are described briefly in the following sections.

7.2.1 Computer Engineer Panel

The computer engineer panel contains 60 neon lights which display in binary form the contents of the computer 12-digit display register which is displayed in decimal form on the operator panel. Switches are provided on the computer panel to operate the display register in either of the three following modes:

(1) In the manual display mode, the engineer can set controls to select and sample, at a specific pulse time and step of an instruction that is being executed, data at various points within the computer and transfer the data to the 12-digit display register where it can be observed in binary form on the engineer panel. Data can be sampled for display from the various registers within the arithmetic and control sections of the computer, from various data transfer paths including the main storage bus, and from various combinations of flip-flops.

(2) In the program display mode, the engineer can, as in the manual display mode, select for sampling, at a specific pulse time and instruction step, data at various points within the computer. However, the display will only be effected for instructions that are tagged with a "one" digit in the most significant digit position.

(3) In the fast register display mode, the engineer can select and sample for display the contents of any of the fast registers within the computer (refer to section 2.5).

The display registers may also be used by the engineer, in conjunction with a type-in from the keyboard, to enter data into the main instruction register of the computer, enter data into any main storage location or display data from any main storage location.

In addition to those associated with the display registers, the following controls and indicators are included on the computer engineer panel.

(1) Error and contingency indicator lights that indicate any error or contingency condition detected by the checking circuits of the computer. The error indicator lights indicate the type of error detected and, for certain types or errors, the digit position at which the error occurred. An error option switch is provided on the panel. The switch has three positions labeled STOP, NORMAL, and IGNORE. When the switch is in the STOP position, the computer stops if an error is detected. When the switch is in the NORMAL position, the computer enters an error routine if an error is detected. When the switch is in the IGNORE position, the computer ignores any error detected. A reset button is also provided with which the engineer may reset the error flip-flops. A similar contingency option switch and a reset button are provided on the panel for controlling contingency conditions.

(2) Two neon pushbuttons are provided on the panel to control the gating in of various control signals that the engineer can manually introduce into the computer for troubleshooting purposes by connecting signal input lines to terminals in the circuits.

(3) A switch abnormal light which is lit if any switch on the computer panel is set in a position to interfere with the running of a normal program or to allow errors to go undetected.

(4) A transfer switch which the engineer may use to force the computer to either transfer control or not transfer control on conditional transfer of control instructions.

(5) Buttons for clearing the computer as a whole or selected parts of the computer.

(6) Retain buttons which can be used to retain the contents of the main instruction register, the control counter, or a fast register.

7.2.2 Processor Engineer Panel

The controls and indicators on the processor panel include:

(1) Illuminated master power control buttons for applying power to the various units of the system. A set of 24 pushbuttons is provided to select the drums that are to be turned on. When the master drum-power control button is pressed, the selected drums are turned on in sequence.

(2) Air-flow, power failure, and temperature indicators for the processor, the computers, each type of input-output device, and each main storage cabinet.

(3) Start and stop pushbuttons.

(4) Illuminated pushbuttons to control the mode in which the processor operates. The processor can operate in a continuous, one instruction, one step, arithmetic test stop, or input-output test stop mode or any of five breakpoint stop modes. With the test stop pushbuttons, the processor can be made to stop on conditional transfer of control instructions (arithmetic tests, input-output tests, or breakpoint tests) and indicate whether a transfer of control is imminent. When the processor stops, the engineer has the option of forcing the processor to transfer control or forcing it to continue on the same sequence of instructions.

(5) Clear buttons for clearing the processor as a whole or selected parts of the processor.

(6) Retain buttons to prevent changing the contents of certain registers within the processor.

(7) Error indicators to display errors detected within the central processor, dispatcher, and each of the synchronizers. Separate error option switches are provided for the central processor and each of the synchronizers.

(8) A switch abnormal light which is lit if any switch on the processor panel is set in a position to interfere with the running of a normal program or to allow errors to go undetected.

(9) A master error-set pushbutton to set all of the processor error flip-flops for the purpose of testing the error flip-flops and the indicators on the panel.

(10) A gain control switch for each tape synchronizer to enable the engineer to set manually the tape amplifier gain control of a tape synchronizer to high or low gain, or have the gain setting determined by the processor program.

(11) A sector-address write switch used in conjunction with a special processor program to lay out the data bands on the magnetic drums when the drums are first installed into the system. If a bad spot (unusable recording area) should develope on a drum after it is installed, one or more bands can be repositioned so that the bad spot will lie in an area that is not used for recording, in a space between sectors, for example.

(12) A memory simulator consisting of a set of 60 switches used to insert manually a word of data into the processor. A word set up on the switches can be directed to the main instruction register, the arithmetic registers and to the synchronizer buffer registers.

(13) Switches to run each of the synchronizers.

(14) A binary display of the main instruction register and various other registers, counters, and control flip-flops in the processor.

(15) Display register controls which can be used to transfer the contents of the main processor instruction register or either arithmetic register to a computer display register for visual inspection, or to transfer the contents of a display register to the instruction and arithmetic registers. Data can be manually entered into the display register from the keyboard.

(16) A type-out button which causes a type-out on a selected console printer of the contents of the processor arithmetic registers.

(17) A jam-signal button which is used to control the gating in of various control signals that may be manually introduced into the processor logical circuits for troubleshooting pruposes.

(18) An audio monitor that can be connected to either computer or to the processor. By changing pitch the audio signal gives an indication of the frequency of instruction execution.

(19) A film monitor to indicate the status of the film in the 35-mm and self-developing cameras of the electronic page printer.

APPENDIX A

COMPUTER INSTRUCTIONS

AND

WORD FORMATS

A-1 WORD FORMATS

A-1.1 Single-Precision Fixed-Point Operands

A single-precision fixed-point operand contains a sign digit and 11 decimal digits, as follows:

S X X X X X X X X X X X

where S is the sign digit and X is a numeric decimal digit. The sign position contains a 0 if the number is positive and a minus digit if the number is negative. An absolute zero is expressed by a period digit in the sign position and all zeros in the X positions. The decimal point is assumed to be between the sign and the most significant digit.

A-1.2 Single-Precision Floating-Point Operands.

A single-precision floating-point operand contains a sign digit, a 2-digit excess-fifty base-ten exponent, and nine decimal digits, as follows:

S Y Y X X X X X X X X X

where S is the sign digit, Y is a floating-point exponent digit, and X is a numeric decimal digit. The decimal point is assumed to be between the exponent (Y) and the most significant (X) digit of the operand number. The operand is normalized, that is, the most significant digit is not zero.

A-1.3 Double-Precision Fixed-Point Operands

A double-precision fixed-point operand consists of two consecutive single precision words each containing a sign digit and 11 decimal digits, as follows:

S X X X X X X X X X X X S X X X X X X X X X X X

where S is the sign digit and X is a numeric decimal digit. The two sign digits must be the same. The decimal point is assumed to be between the first sign digit and the most significant digit of the operand.

A-1.4 Double-Precision Floating-Point Operands

A double-precision floating-point operand consists of two consecutive single-precision words. The first word contains the sign digit, a 2 digit excess-fifty base-ten exponent, and nine decimal digits; the second word contains the sign digit and 11 decimal digits, as follows:

S Y Y X X X X X X X X S X X X X X X X X X X

where S is the sign digit, Y is a floating-point exponent digit, and X is a numeric decimal digit. The decimal point is assumed to be between the exponent (Y) and the most significant (X) digit of the operand number. The operand is normalized, that is, the most significant digit is not zero.

A-1.5 Instructions

A computer instruction word consists of 12 decimal digits, as follows:

T I I A A B B M M M M M

The meanings of the characters that are used to represent the different parts of the instruction word are:

T - tracing mode (refer to section 2.6). If an instruction is to be traced, T should be a digit from 1 to 9 depending upon which tracing mode routine the programmer wishes to specify. If an instruction is not to be traced, T should be a period digit. Any other character in this position will cause a transfer of control to an error routine.

I - instruction designator. I specifies the operations to be executed by an instruction. An instruction designator that is not in the computer repertory of instructions will cause a transfer of control to an error routine.

A - arithmetic register address. A normally specifies the address of one of a number of fast register (refer to section 2.5) that are used as a arithmetic registers for storing operands and the results of operations.

B - B-register address. B normally specifies the address of one of a number of fast registers that are used as index (B) registers for storing an address modifier. The M address of an instruction is automatically modified by a modifier, Δ, in the specified index register after the instruction is read from storage but before the instruction is executed. If the index register specified is 00, the M address is not modified. All normal computer instructions containing an operand address may refer to an index register for modification of the address.

M - memory address. M normally specifies (after it is modified by the modifier in a specified index register) the address of an operand. M may specify a main storage address or the address of a fast register (refer to section 2.5).

A-1.6 Index Words

A fast register when it is used as an index (B) register contains a 12-digit word, as follows:

N N N D D D D Δ Δ Δ Δ Δ

The meanings of the characters that are used to signify the different parts of the index word are:

N - cycle count. N specifies the number of times an iterative program loop is to be traversed. N is reduced by one each time the program loop is traversed. When the count becomes zero the iterative process is terminated (refer to index instructions table A-1).

D - decrement or increment to Δ. Before each program loop traversal D is added to or subtracted from Δ. It therefore determines the amount by which the operand addresses of indexed instructions are modified during each program loop traversal relative to the previous loop traversal.

Δ - modifier. Δ is automatically added to the M part of an instruction that addresses the index register.

A-2 INSTRUCTIONS

A-2.1 Conventions

The following conventions are used in presenting the computer instructions (table A-1).

A - denotes the number of a fast register. The next fast register in sequence is denoted as A+1, and the preceding fast register as A-1.

M - denotes the number of a particular location (12-digit) in the main storage. The next storage location in sequence is denoted as M+1, and the preceding location as M-1.

A_A - a capital letter subscript denotes a particular part of a fast register or storage location in accordance with the instruction word format. For example:

A_A - denotes the A-register address part of fast register A.

A_B - denotes the B-register address part of fast register A.

A_I - denotes the instruction designator part of fast register A.

A_M - denotes the memory address part of fast register A.

M_A, M_B, M_I, and M_M - denote, respectively the A-register address, B-register address, instruction designator, and memory address parts of storage location M.

$A_{A,B}$ - more than one part of a fast register or storage location may be denoted by successive capital letter subscripts. For example:

$A_{A,B}$ - denotes the A-register address and B-register address parts of fast register A.

FFA - denotes the address of a flip-flop, such as a tracing mode, sense, or error flip-flop. The address of the flip-flop is specified in the A-part of an instruction.

C - denotes a control counter register which can be assumed to contain the main storage address of the instruction currently being executed.

( ) - parentheses around a symbol denotes the contents of the control counter, register, storage location, or part of a register or storage location which is indicated by the symbol. For example:

(C) - denotes the contents of the control counter.

(M) - denotes the contents of main storage location M.

(A_B) - denotes the contents of the B-register address part of fast register A.

(C)+1-->C- denotes that the present sequence of executing instructions is continued, that is, one is added to contents of the control counter register to give the storage address of the next instruction in sequence.

M-->C - denotes that control is transferred to a new sequence on instructions starting with the instruction in storage location M.

rdd - denotes a rounded result. All other results are unrounded.

◯ - a circled arithmetic symbol denotes a floating-point arithmetic operation. For example:

(M)⊕(A) - denotes floating point addition of the contents of storage location M and fast register A.

| | - two parallel vertical lines around a quantity denote the absolute value of that quantity. Thus:

|(A)| - denotes the absolute value of the contents of fast register A.

' - A prime denotes a double-precision word. Thus:

(A') - signifies two single-precision words in successive fast registers (A and A+1).

NOTE

Although M is usually an address of a location within the main ferrite-core storage, it may be an address of a fast register. The main storage is assigned M-addresses from 00000 to 97499. The fast registers are assigned M-addresses from 99901 to 99999.

A-2.2 Instruction List

The computer instructions are listed in table A-1. The numeric operation code (instruction designator) is given for each instruction together with a mnemonic code and a symbolic description of the instruction using the conventions described above. The execution time of each instruction is given in microseconds. The execution times are all inclusive and include the time required for obtaining instructions and operands from storage, the time required to modify operand addresses, the time required to calculate floating-point exponents, the time required for error, contingency, and tracing mode checking, and so forth. All input-output operations may be assumed to be performed in parallel with the instructions.

Table A-1. Computer Instructions

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.	Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
		Data Transfer Instructions (To Storage)					
40	S	(A) ---> M	4	45	SS	(A') --> M'	8
41	SM	-(A)---> M	4	46	SSN	-(A')--> M'	8
42	SM	|(A)| -> M	4	47	SSM	|(A')| --> M'	8
		Data Transfer Instructions (From Storage)					
43	F	(M) ---> A	4	62	EB	(M_B) --> A_B	4
48	FF	(M') --->A'	8	63	EAB	(M_{AB})---> A_{AB}	4
60	EOP	(M_I)---> A_I	4	64	EM	(M_M) ---> A_M	4
61	EA	(M_A) --->A_A	4				
		Floating-Point Add Instructions					
02	A	(M)⊕(A)-->A	4	14	NU	-(M)⊕(A)-> A+1	4
03	AM	|(M)|⊕(A)-->A	4	06	AA	(M')⊕(A')-->A'	16
04	AU	(M)⊕(A)--->A+1	4	16	NN	-(M')⊕(A')->A'	16
12	N	-(M)⊕(A)-->A	4				

Table A-1. (continued)

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.	Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
		Floating-Point Multiply Instructions					
23	M	(M)⊗(A)->A	8	27	MM	(M')⊗(A')-->A'	36
24	MU	(M)⊗(A)-->A+1	8	22	MR	(M)⊗(A)Rdd->A	12
25	ME	(M)⊗(A)->A'	12				
		Floating-Point Divide Instructions					
32	DR	(A)⊘(M)Rdd->A	28	37	DSE	(A')⊘(M)-->A'	60
34	DUR	(A)⊘(M)Rdd-->A+1	28	36	DD	(A')⊘(M')-->A'	168
		Fixed-Point Add Instructions					
01	AX	(M)+(A)-->A	4	05	AAX	(M')+(A')-->A'	12
11	NX	-(M)+(A)->A	4	15	NNX	-(M')+(A')->A'	12
		Fixed-Point Multiply Instructions					
20	MXR	(M)X(A)Rdd->A	8	26	MMX	(M')X(A')->A'	36
21	MXE	(M)X(A)---->A'	12				
		Fixed-Point Divide Instructions					
30	DX	(A)÷(M)->A	32	35	DDX	(A')÷(M')->A'	184
31	DXE	(A)÷(M)->A'	36				

Table A-1. (continued)

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.	Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
Conditional Transfer of Control Instructions							
71	TG	(A)>(A+1)? No:(C)+1-->C Yes:M --->C	 4 12	74	TLZ	(A)<0? No:(C)+1--> C Yes: M ---->C	 4 12
70	TE	(A)=(A+1)? No:(C)+1->C Yes: M --->C	 4 12	76	TTG	(A')>(A+2)' No:(C)+1--->C Yes: M ---->C	 8 16
73	TGZ	(A)>0? No:(C)+1-->C Yes: M---->C	 4 12	75	TTE	(A')=(A+2)' No:(C)+1--->C Yes: M ---->C	 8 16
72	TZ	(A)=0? No:(C)+1-->C Yes: M -->C	 4 12	95	TF	Test FFA FFA reset: (C)+1--->C FFA set: M ----> C	 4 12
Unconditional Transfer of Control Instructions							
90	T	M -----> C	8	92	TB	(C) --> A_M and M -----> C	8
91	TR	90(C)+1-->M and M + 1 -->C	 12				

Instruction 91(TR) automatically jams a 90(T) instruction, with (C)+1 in its five least significant digit positions (the M-part), into storage location M and transfers control to the instruction in M+1 which is the beginning of a subroutine. An instruction at the end of the subroutine transfers control to the 90 instruction in storage location M which, in turn, returns control to the next instruction, (C)+1, in the originating program.

Instruction 92 stores the contents of the control counter (the address of this instruction) in the five least significant digits (the M-part) of fast register A and transfers control to the instruction in storage location M which is the beginning of a subroutine. Fast register A is used as an index register to modify an exit instruction at the end of the subroutine so that it will return control to the originating program. The exit instruction is a 90(T) instruction in the form: T90 00 AA 00001. This instruction, since it is modified by the M-part of fast register A, transfers control to (C)+1.

Table A-1. (continued)

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.	Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
			Shift Instructions				
52	PR	$(A)10^{-M}\rightarrow A$	4	58	PPL	$(A')10^{M}\rightarrow A'$	8
57	PPR	$(A')10^{-M}\rightarrow A'$	8	59	PPC	Left circular shift (A') M places	12
53	PL	$(A)10^{M}\rightarrow A$	4				
			Conversion Instructions				

Instruction 51 converts a fixed-point single-precision number in fast register A to a floating-point single-precision number which is stored in fast register A. Instruction 50 converts to a floating-point single-precision number in fast register A to a fixed-point single-precision number which is stored in fast register A. The conversions are made in accordance with a scale factor specified by the two least significant digits of the instruction.

Instructions 56 and 55, are the same as instructions 51 and 50 respectively, except that the conversions are made to and from double precision words in fast registers A and A + 1.

51	C	FX--->FL M=scale factor	4	56	CC	FX'--->FL' M=scale factor	12
50	CX	FL--->FX M=scale factor	4	55	CCX	FL'--->FX' M=scale factor	12
			Extract Instructions				

Instruction 65 erases certain digits in fast register A and replaces them with digits extracted from corresponding positions of fast register A - 1. Digits are extracted only in the digit positions corresponding to digit positions occupied by the decimal number 1 in storage location M. Either a 1 or a - causes extraction in the sign position.

Instruction 66 is the same as 65 except that the digits are extracted from fast register A + 1 instead of A - 1.

65	EL	(A-1)->A (M)	8	66	EU	(A+1)->A (M)	8

Table A-1. (continued)

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.	Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
		Index Instructions					
		The characters N, D, and Δ signify the parts of a word stored in fast register A which is used as an index register. The index word format is described in the first part of this appendix.					
80	BIT	N-1---> N		81	BDT	N-1---> N	
		If New N=0, (C)+1-->C	12			If New N=0, (C)+1-->C	12
		If New N≠0, M ---> C and Δ+D--> D	8			If New N≠0, M ---> C and Δ -D--> Δ	4
82	BIC	N-1--->N		83	BDC	N-1--->N	
		If New N=0, M --> C	12			If New N=0, M -> C	12
		If New N≠0, Δ+D--> Δ and (C)+1-->C	8			If New N≠0, Δ-D--> Δ and (C)+1--->C	4
85	BI	Δ + D --> Δ	4	86	BD	Δ - D --> Δ	4
		Miscellaneous Instructions					
97	SF	Set FFA	4	00		Skip	4
96	RF	Reset FFA	4	99	H	Stop	

Table A-1. (continued)

Numeric Code	Mnemonic Code	Symbolic Description	Time μsec.
		The interlock and connect flip-flops are set by the operator when he enters data into a display register from a keyboard. Instructions 09 and 19 test the interlock flip-flop to ensure that only data entered into a display register by the operator are picked up.	
		Instructions 29 and 39 test the interlock flip-flop to ensure that data are not transferred into a display register by the computer while the operator is entering data for the computer or processor program by way of the display register.	
09	FV	If interlock ff set, (5-digit display register)----> A_M and connect and interlock ff's are reset	4
		If interlock ff reset, M---> C	12
19	FVK	If interlock ff set, (12-digit display register)----> A and connect and interlock ff's are reset.	4
		If interlock ff reset M---> C	12
29	SV	If interlock ff reset, (A_M)--> 5-digit display register.	4
		If interlock ff set, M----> C	12
39	SVK	If interlock ff reset, (A)--> 12-digit display register.	4
		If interlock ff set, M----> C	12

APPENDIX B

CHARACTER CODES

B-1 GENERAL

This appendix briefly describes the basic character codes used in the LARC system and the way in which they are represented on the various input-output devices. A comparison is made in table B-2 between the codes and the characters or functions they represent on each of the input-output devices. If an output device can, depending upon the mode in which it is operating, decode a code combination to either perform a function or print a character, this is indicated in table B-2 by listing an abbreviation for the function followed by the character, in parentheses. The abbreviations and their meanings are listed at the bottom of the table.

B-2 UNIVAC CODE

LARC data is represented on magnetic tape in the UNIVAC seven-bit code (table B-2). Input information from the tapes is automatically translated to a LARC internal code and output information that is to be recorded on tapes is automatically translated back to UNIVAC code. The tape read-write synchronizers in the processor can be instructed to translate a block of numeric input data into the LARC one-digit numeric code, or translate a block of alphanumeric data into the LARC two-digit alphanumeric code. Similarly output data can be translated from either the LARC one-digit numeric code or two-digit alphanumeric code. Tapes may be manually prepared using a Unityper I, Unityper II, or a Tape Verifier all of which record on Univac code.

B-3 LARC ONE-DIGIT NUMERIC CODE

The basic internal code of the LARC is a five-bit biquinary code in binary-coded decimal form. Fifteen digit combinations are allowed, any one of which may be stored in any digit position of the storage. The 15 combinations and the characters they normally represent are shown in table B-1. The fifth bit of the code is an odd parity check bit. Only combinations containing an odd number of "ones" are allowed. The code combination 01101, although it contains an odd number of "ones", is not allowed.

In the computer, all 15 digit combinations can be shifted, extracted or transferred. Except for a minus or a period digit in the sign position, only the numerics 0 through 9 are allowed in the adder-comparator of the computer. In the processor adder-comparator, however, the non-numeric digits (plus, minus, space, period, and ignore) are allowed in any digit position. When non-numerics are added, they appear in the result in accordance with a predetermined order of precedence.

Table B-1. LARC One-Digit Numeric Code

Bit Positions	
5 4 3 2 1	Character
1 1 1 0 0	\ (ignore)
0 0 1 0 0	Λ (space)
0 0 0 1 0	— (minus)
1 0 0 0 0	0
0 0 0 0 1	1
1 0 0 1 1	2
0 0 1 1 1	3
1 0 1 1 0	4
0 1 0 0 0	5
1 1 0 0 1	6
0 1 0 1 1	7
1 1 1 1 1	8
0 1 1 1 0	9
1 1 0 1 0	. (period)
1 0 1 0 1	+ (plus)

B-4 LARC TWO-DIGIT ALPHANUMERIC CODE

An alphanumeric character is represented in LARC by two adjacent numeric digits that are handled as a pair. An equivalent alphanumeric character is represented on magnetic tape by a single seven-bit digit in UNIVAC code. The decimal equivalent of the LARC two-digit code is shown in table B-2.

B-5 ELECTRONIC PAGE PRINTER

The electronic page printer synchronizer can operate in any one of the four following modes:

(1) Numeric edited.

(2) Numeric unedited.

(3) Alphanumeric edited.

(4) Alphanumeric unedited.

When the synchronizer operates in the numeric modes (modes 1 and 2), a single LARC digit is decoded to perform a function or print a character. In the numeric edited mode, the ignore digit (11100) and the space digit (00100) are decoded to perform specific functions. The ignore digit performs an end-of-word function. For example, a 12-digit word containing an ignore digit in the ninth digit position is shortened to eight digits when it is printed in the numeric edited mode. The space digit performs a space function, that is, it leaves a single space in the printed copy. In the numeric unedited mode the ignore and space digits are decoded to print as \ and ∧, respectively.

When the synchronizer operates in the alphanumeric modes (modes 3 and 4), a LARC two-digit alphanumeric combination is decoded to perform a function, or print a character. In the alphanumeric edited mode, the combinations 15, 16, and, 35 are decoded to perform specific functions. A 15 performs an ignore function, that is, the synchronizer neither prints a character nor leaves a space but ignores the combination. A 16 performs a space function. A 35 performs an end-of-ten-words function, that is, no more data are sent to the printer. In the alphanumeric unedited mode, the two-digit combinations 15, 16 and 35 are decoded to print as \ , ∧, and ϵ, respectively.

B-6 ON-LINE PRINTER

The on-line printer synchronizer can operate in any one of the four following modes:

(1) Numeric edited.

(2) Numeric unedited.

(3) Alphanumeric edited.

(4) Alphanumeric unedited.

When the printer operates in the numeric modes (modes 1 and 2), a single LARC digit is decoded to perform a function or print a character. In the numeric edited mode, both the ignore (11100) and the space (00100) digits are decoded to leave a space. In the numeric unedited mode, the ignore and space digits are decoded to print as I and Δ, respectively.

When the synchronizer operates in the alphanumeric modes (modes 3 and 4), a LARC two-digit alphanumeric combination is decoded to perform a function or print a character. In the alphanumeric edited mode, a 16 combination is decoded to leave a space. In the alphanumeric unedited mode, a 16 combination is decoded to print as a W.

B-7 CONSOLE PRINTER.

Because the console printer is a relatively slow device, its synchronizer is designed to operate only in the alphanumeric mode. The form a print-out takes is determined by the processor program. If numeric data is printed, it is first translated by the processor program to the two-digit alphanumeric code. The printer synchronizer decodes a LARC two-digit alphanumeric combination to perform a function or print a character. Many of the two-digit combinations can print either of two characters depending upon whether the type basket of the printer is in the upper or lower case. The type basket is shifted to the upper case position by the two-digit combination 10 and is shifted to the lower case position by the combination 11.

The following two-digit combinations are decoded by the synchronizer to perform specific functions. In each case the same function is performed regardless of whether the type basket is in the upper or lower case.

15 - neither prints a character nor leaves a space, but is ignored.

16 - leaves a space.

35 - returns the carriage to the left margin.

55 - advances the carriage to the next preset tab stop.

The console printer may also be used to prepare punched paper tapes. With the type basket in the upper case position, all 15 LARC code combinations and several special code combinations may be punched on the paper tape.

B-8 CONSOLE KEYBOARD

The keyboard on the operator control console is used to manually enter data into either the five-digit or 12-digit display register. It consists of 18 keys. Fifteen keys are used to enter the 15 LARC code combinations into either display register, two keys are used to connect the keyboard to the proper display register and one key is used to disconnect the keyboard from the display registers.

B-9 CONSOLE DECIMAL DISPLAYS

The decimal displays on the operator control console display in decimal form the contents of the five-digit display register, the 12-digit display register, and a computer control counter register. Characters representing all fifteen LARC code combinations may be displayed. Since the number of single characters that can be displayed is limited to 12, the ignore, space, and period are represented by superimposing one character upon another. The ignore is represented by a 1 and 8 superimposed, the space by a 0 and — superimposed, and the period by a 1 and 0 superimposed.

Table B-2. Comparison of LARC Character Codes

UNIVAC Code on Tape	LARC Num. 1-Digit Code	LARC Alpha. 2-Digit Code	Standard Unityper II	Electronic Page Printer Num. Mode	Electronic Page Printer Alpha. Mode	On-line Printer Num. Mode	On-line Printer Alpha. Mode	Console Printer (Alpha.) Upper Case	Console Printer (Alpha.) Lower Case	Console Keyboard (Num.)	Console Decimal Displays (Num.)
		10						UC	UC		
		11						LC	LC		
		2-						-	-		
		2^						^	π		
		2+						+	+		
		2.						.	.		
		2&						/	&		
1 00 0000	11100	15	i	ECW (&)	IG (&)	SP (I)		IG	IG	/	⊕
0 00 0001	00100	16	^ .	SP(^)	SP (^)	SP (▲)	SP (W)	SP	SP	^	⊖
0 00 0010	00010	17	-	-	-	-	-	-	-	-	-
1 00 0011	10000	20	0	0	0	0	0	0	λ	0	0
0 00 0100	00001	21	1	1	1	1	1	1	U	1	1
1 00 0101	10011	22	2	2	2	2	2	2	I	2	2
1 00 0110	00111	23	3	3	3	3	3	3	O	3	3
0 00 0111	10110	24	4	4	4	4	4	4	J	4	4
0 00 1000	01000	25	5	5	5	5	5	5	K	5	5
1 00 1001	11001	26	6	6	6	6	6	6	L	6	6
1 00 1010	01011	27	7	7	7	7	7	7	M	7	7
0 00 1011	11111	28	8	8	8	8	8	8	,	8	8
1 00 1100	01110	29	9	9	9	9	9	9	/	9	9
0 00 1101		32	'	=	=		=		=		
0 00 1110		33	&	~	~		~	€	~		
1 00 1111		34	(	(	(		(		(		
0 01 0000		35	r	ECL (€)				CR	CR		
1 01 0001		36						8			
1 01 0010	11010	37	, .	.	, .	.	, .	.	, .	.	⊙
0 01 0011		40	;		▽		▽		▽		
1 01 0100		41	A		A		A		A		
0 01 0101		42	B		B		B		B		
0 01 0110		43	C		C		C		C		
1 01 0111		44	D		D		D		D		
1 01 1000		45	E		E		E		E		
0 01 1001		46	F		F		F		F		
0 01 1010		47	G		G		G		G		
1 01 1011		48	H		H		H		H		
0 01 1100		49	I		I		I		I		
1 01 1101		52	#		<		<	> *	<		
1 01 1110		53	¢		π			→	π		
0 01 1111		54	@		←			≤	←		
0 10 0000		55	t		]			:	]		
1 10 0001		56	"		[			2	[		
1 10 0010		57	—		)		)		)		
0 10 0011		60	)		J		J	^	J		
1 10 0100		61	J		K		K	TAB	K		
0 10 0101		62	K		L		L	◁	L		
0 10 0110		63	L		M		M	□	M		
1 10 0111		64	M		N		N	4	N		
1 10 1000		65	N		O		O	5	O		
0 10 1001		66	O		P		P	6	P		
0 10 1010		67	P		Q		Q	7	Q		
1 10 1011		68	Q		R		R		R		
0 10 1100		69	R		> *		> *	3			
1 10 1101		72	$		→						
1 10 1110		73	*		≤						
0 10 1111		74	?		←						
1 11 0000		75	Σ		:		:				
0 11 0001		76	∂		+		+				
0 11 0010		77	:		/		/				
1 11 0011	10101	80	+	+	S	+	S	+	+	+	+
0 11 0100		81	/		T		T	9	/		
1 11 0110		82	S		U		U		S		
1 11 0110		83	T		V		V		T		
0 11 0111		84	U		W		W	1	U		
0 11 1000		85	V		X		X		V		
1 11 1001		86	W		Y		Y		W		
1 11 1010		87	X		Z		Z		X		
0 11 1011		88	Y		Δ		Δ		Y		
1 11 1100		89	Z		λ				Z		
0 11 1110		92	%		□				Δ		
0 11 1110		93	=					0	λ		
1 11 1111		94									

Abbreviations

ECW - End of Word
SP - Space
IG - Ignore
ECL - End of Line (10-Words)
UC - Upper Case
LC - Lower Case
CR - Carriage Return
TAB - Tabulate

APPENDIX C

SUMMARY ORDERS FOR A PROCESSOR PROGRAM

C-1 DISCLOSURE WORDS

C-1.1 General

The computer program notifies the processor program that a group of summary orders are ready for execution by depositing a disclosure word in a predetermined storage location (2500) and setting a disclosure flip-flop. The processor program tests the disclosure flip-flop and, if it is set, picks up the disclosure word and resets the flip-flop. The disclosure word discloses to the processor program the location of one or more summary orders stored in sequential storage locations. Such a group of summary orders is called a packet. A packet of summary orders also contains an end-of-packet word which has a count of the number of summary orders in the packet that call for a transfer of data to or from the main storage. A disclosure word may take one of the three forms described below.

C-1.2 Primary-Program Disclosure Word

A primary-program disclosure word is used most often and is issued by the main program being run on the computer. It has the following format:

15 NNNNN MMMMM

where N is the storage address of the first summary order of the packet being disclosed and M is the storage address of the end-of-packet word. The digits 15 are operation digits which instruct the processor program to record the time at which the disclosure word is received and inform the operator by means of a print-out on the console printer if another primary-program disclosure word is not received within five minutes. The print-out is repeated at five minute intervals until the processor program receives a new primary-program disclosure word. The print-out serves to alert the operator in the event a computer program gets stuck in an endless loop which prevents the primary program from issuing summary orders.

C-1.3 Secondary-Program Disclosure Word

A secondary-program disclosure word is issued by a secondary or side program being run on the computer. It has the following format:

14 NNNNN MMMMM

This disclosure word is interpreted by the processor program in the same way as a primary-program disclosure word except that only one print-out is made and the print-out indicates that a secondary-program disclosure word was not received within the past five minutes.

C-1.4 Pseudo Disclosure Word

A pseudo disclosure word is provided which consists of all zeros, as follows:

000000000000

A pseudo disclosure word is issued in certain instances by the computer program as an expedient in determining when the processor program has picked up the last packet of summary orders so that they may be modified and reissued. After the processor program picks up a disclosure word, it resets the disclosure flip-flop. It does not thereafter test the disclosure flip-flop until all of the summary orders disclosed have been picked up and either started in execution or filed in its program. The computer program is not aware that all of the summary orders disclosed by one disclosure word have been picked up until the processor program resets the disclosure flip-flop to acknowledge the receipt of the next disclosure word. When the computer program tests the disclosure flip-flop and finds it reset, it is then assured that the summary orders disclosed by the next to last disclosure word have been picked up and can therefore be modified. Instead of making a real disclosure of summary orders, the computer may determine if the last packet of summary orders have been picked up by issuing a pseudo disclosure word. The only function of the pseudo disclosure word is to force the processor program to reset the disclosure flip-flop after it tests it and by so doing inform the computer program that the previous packet of summary orders have been picked up. The pseudo disclosure word is otherwise ignored by the processor program.

C-2 END-OF-PACKET WORDS

C-2.1 General

An end-of-packet word is always filed with a packet of summary orders that the computer program discloses to the processor program. The end-of-packet word contains a count of the number of summary orders that call for a transfer of data to or from the main storage. Each time the processor completes one of these summary orders, it lowers the count by one. By checking the count for zero, it can be determined when all of the storage transfers specified in the packet have been completed and the computer program is free to use the storage areas involved. An end-of-packet word can take either of two forms; one form places the burden of checking for completion on the computer program; the other form places the burden of checking on the processor program.

C-2.2 Computer-Check End-of-Packet Word

A computer-check end-of-packet word has the following format:

000 NN 00 00000

where NN is a count of the number of summary orders in the packet that call for a transfer of data to or from the main storage. The processor program lowers the count by one each time one of these summary order is completed. The computer program checks the count for zero to determine when all storage-transfer summary orders are completed.

C-2.3 Processor-Check End-of-Packet Word

A processor-check end-of-packet word has the following form:

.90 NN 00 MMMMM

where NN is a count of the number of summary orders in the packet that call for a transfer of data to or from the main storage. The processor program lowers the count by one each time one of these summary orders is completed and also checks the count for zero. When the count reaches zero, the processor

program places a .90 0000 MMMMM computer instruction in a processor intervention routine and sets the processor intervention flip-flop. Setting the intervention flip-flop interrupts the computer program and forces it to enter the processor intervention routine where the .90 0000 MMMMM instruction is executed. This instruction is an unconditional transfer of control instruction which transfers control to a subroutine beginning at MMMMM. The intervention is programmed in such a way that control can be transferred back to the point of interruption in the main program after the subroutine is completed.

C-3 SUMMARY ORDERS

The summary orders for a processor control program that is furnished with the LARC system are described in table C-1.

It should be understood that summary orders are not processor instructions that are decoded by a built-in decoder in the processor. Rather, they are pseudo or program orders for a specific processor input-output control program which receives and interprets the summary orders and, on the basis of this interpretation, executes actual processor instructions at the proper time and in the proper order and sequence.

It should also be understood that other processor input-output control programs can be designed which accept different sets of summary orders or interpret the summary orders differently or which execute input-output operations for specific types of computer problems without receiving summary orders.

The format of a summary order varies depending upon the type of summary order. However, the two most significant digits are usually operation digits which specify what operations are to be performed by the order and the remaining digits usually specify the input-output device or storage area to be used or the extent of the operation, or both. The format of each summary order is listed in table C-1, along with a mnemonic name for the summary order and a description of how the processor input-output control program interprets the summary order. A lower case x shown in a digit position of the summary order format indicates that the digit position is ignored by the processor program, however, it should contain one of the digits 0 through 9. A digit position shown as a 0, must contain a zero for the summary order to be interpreted correctly. Other digit positions are symbolized in table C-1 by characters whose meanings are defined in the descriptions of the summary orders.

Table C-1. Summary Orders

Format	Name	Interpretation by Processor Control Program
00 xxxxx xxxxx	Skip	Ignore
03 TTTTT MMMMM	Time Limit	When TTTTT seconds have elapsed transfer computer control to the instruction in storage location MMMMM.
06 S xxxxx MMMMM	Stop	If S = 0, accept no more summary orders from the computer; complete all summary orders already accepted; transfer computer control to the instruction in storage location MMMMM; and, after computer control has been transferred, begin again to accept and execute summary orders from the computer. Note: A summary order of this type is normally issued when the computer switches over to a new problem. If S = 1, stop executing summary orders; transfer a fresh processor program to the core storage; transfer computer control to the instruction in storage location MMMMM; and reset the disclosure flip-flop. Note: A summary order of this type is normally issued in the event of an untenable mix-up in the computer or processor program, or both. If S = 2, complete all summary orders currently being executed; transfer the processor program currently in the core storage to a service drum; substitute a fresh processor program; and transfer computer control to the instruction in storage location MMMMM. Note: A summary order of this type is normally issued in the event there is a mistake in the computer program. In order to analyze the program, the programmer might desire a print-out of data in the core storage. Consequently, all summary orders that are in the process of being executed are completed first so that any transfers of data to the core storage that are in progress will be completed before a print-out is initiated.

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
08 xxxxx MMMMM	Storage Unit Write Interlock	Execute an instruction which interlocks the computer program against writing data directly into the 2500-word storage unit containing storage location MMMMM. Do not as a result of this summary order interfere with the execution of summary orders which specify that data be transferred to the same storage unit. Note: If the computer program attempts to write data directly into an interlocked storage unit, computer control is transferred to an error routine.
09 xxxxx MMMMM	Remove Storage Unit Write Interlock	Execute an instruction which removes the interlock against the computer program writing into the storage unit containing storage location MMMMM. If the processor program is stored in the same storage unit, do not interrupt the computer program but notify the operator by means of a printout at the operator control console.
10 C x TTT MMMMM	Visual Display	If C = 1, transfer the contents of storage location MMMMM to the 12 digit display register where it may be observed in decimal form at the operator control console and in binary form at the engineer control console. Time the display for a minimum of TTT seconds so that the operator will notice it. If C = 2, transfer the contents of the five least significant digit positions of storage location MMMMM to the five-digit display register where it may be observed in decimal form at the operator control console and in binary form at the engineer control console. Time the display for a minimum of TTT seconds so that the operator will notice it.
18 NNNNN MMMMM	Interlock Storage for Editing	Delay the execution of any summary orders from the computer which specify a transfer of data to or from the storage area beginning with storage location NNNNN and ending with storage location MMMMM until the interlock is removed. Note: The specified storage area is reserved for summary orders from an editing routine that is controlled by the processor. The interlock can be removed only by the processor editing routine.

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
19 NNNNN MMMMM	Edit	The storage area beginning with storage location NNNNN and ending with storage location MMMMM contains information for the editing routine.
20 nn D W m MMMMM	Console Printer Print	Return the carriage, then print nn words of data in print mode m on console printer numbe D beginning with the word stored in storage location MMMMM. After every W(1-9) words return the carriage except in mode 3 (see note below). Note: The following printing modes may be specified: m = 1 - numeric edited m = 2 - numeric unedited m = 3 - alphanumeric edited m = 4 - alphanumeric unedited When printing in mode 3, carriage returns are specified by digits within the data words themselves.
28 NNNNN MMMMM	Summary Order Storage Interlock	If a summary order is received which would alter the contents of the storage area beginning with storage location NNNNN and ending with storage location MMMMM, do not execute. Instead, transfer computer control to the instruction in storage location NNNNN. Note: Only one storage area may be interlocked in this fashion at any one time. To remove the interlock, it is necessary to issue another 28 summary order.
Line Printer Summary Orders		
40 nn S Q m MMMMM	Print	Print nn X 10 words of data in print mode m on line printer number S beginning with the word stored in storage location MMMMM. Advance the paper Q lines (Q must be > 0) before each line is printed. Note: Q = 1 for single spacing, Q = 2 for double spacing, etc. The following print modes may be specified. m = 1 - numeric edited m = 2 - numeric unedited m = 3 - alphanumeric edited m = 4 - alphanumeric unedited

Table C-1. Summary Order (cont.)

Format	Name	Interpretation by Processor Control Program
42 LL S xx xxxx C	Advance Paper	If C = 0, advance the paper LL lines on line printer number S. If C = 1, advance the paper on line printer number S to the top of the next page, then advance it LL lines.
43 PP S SL xxx FL	Page Format	The page format of line printer number S consists of PP lines (normal size paper is 66 lines). Printing shall begin on each page after advancing the paper SL lines and end on line FL (FL>SL).
	Electronic Page Printer Summary Orders	
50 BBB W m MMMMM	Print or Plot	Print or plot BBB X 10 words of data in mode m beginning with the word stored in storage location MMMMM. Print W X 10 words per line. Do not advance the film or reposition the beam on the completion of this summary order. Note: The film may be advanced and the beam repositioned while this summary order is being executed as a result of the last 53 summary order to be executed. The following modes may be specified. If mode 5, 6, 7, or 8 is specified, W must equal 0. m = 1 - numeric edited m = 2 - numeric unedited m = 3 - alphanumeric edited m = 4 - alphanumeric unedited m = 5 - graphing with the X and Y coordinates for two points contained in each LARC word, as follows: XXX YYY X'X'X' Y'Y'Y' m = 6 - graphing with the X and Y coordinates for a single point contained in two successive LARC words as follows: xxx XXX xxxxxx xxx YYY xxxxxx m = 7 - plotting vertical grid lines with two abscissas for two full length vertical grid lines contained in each LARC word, as follows: xxx HHH xxx H'H'H'

Table C-1. Summary Orders (cont.)

Format		Interpretation by Processor Control Program
		m = 8 - plotting horizontal grid lines with two ordinates for two full length horizontal grid lines contained in each LARC word, as follows: xxx HHH xxx H'H'H'
51 BBB W m MMMMM	Print and Advance or Plot and Advance	Same as summary order 50 above except that upon completion of this summary order advance the film and reposition the beam to X = 000, Y = 999.

S, Z, L, X or Y in summary orders 52, 53, and 56 must be expressed as a number of points in a 1000 x 1000 point mesh; where 15 such points are equal to one normal line spacing.

52 SSS xxxx ZZZ	Line Spacing	Execute SSS spaces or change spacing between lines to ZZZ or both.
53 x LLL XXX YYY	Position	Position beam to point XXX, YYY. When line LLL has been printed, advance film and again position beam to XXX, YYY.
56 xxxxxxxx KK	Select Plotting Character	Select plotting character KK for use in the plot summary orders to follow.
57 xxxxx xxxxx	Open Shutter	Open the Polaroid Land camera shutter of the printer connected to the synchronizer.
58 xxxxx xxxxx	Close Shutter	Close the Polaroid Land camera shutter of the printer connected to the synchronizer.
59 xxxx C xxxxx	Connect	Connect printer C to the printer synchronizer.

Magnetic Tape Summary Orders

The block length mode for a given tape can be set to either fixed or variable. Other tape summary orders thereafter issued for that tape (such as read or position summary orders) are interpreted in accordance with the mode specified by the 63 or 64 summary order.

63 K L x ST x WWW0	Fixed Block	The length of the blocks on tape ST will be

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
		WWW0 words in length. Indicate an error whenever BBB in summary order 73, 74, 75, or 77 is not a multiple of WWW. If L = 0, the space between blocks on tape ST will be 1 inch. If L = 1, the space between blocks on tape ST will be 2.4 inches. If K = 1, data read from tape ST will be translated from UNIVAC code to LARC one-digit numeric code and data to be written on tape ST will be translated from LARC one-digit numeric code to UNIVAC code. If K = 2, data read from tape ST will be translated from UNIVAC code to LARC two-digit alphanumeric code and data to be written on tape ST will be translated from LARC two-digit alphanumeric code to UNIVAC code.
64 K L x ST xxxxx	Variable Block Length Format	The blocks on tape ST will be of different lengths, but will always be a multiple of ten words. K and L are interpreted in the same way as they are by summary order 63.

Unless otherwise indicated, the following tape summary orders are interpreted in the same way for both the fixed or variable tape modes.

66 xxx ST xxxxx	Rewind	Rewind tape ST.
58 xxx ST xxxxx	Rewind with Interlock	Rewind tape ST and interlock the tape unit against reading or writing until the operator releases the tape unit from the interlock.
71 C D x ST EEEEE	Position Forward	Position tape ST forward through EEEEE blocks. If C = 0, check to insure that each block of data is readable. If C = 1, do not check the readability of the blocks. If D = 0, use synchronizer S for positioning. If D = 1, use synchronizer 9 (rewind checker) for positioning.
72 C D x ST EEEEE	Position Backward	Position tape ST backward through EEEEE blocks. C and D are interpreted in the same way as

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
		they are by summary order 71.
73 BBB ST MMMMM	Read Forward	Fixed Mode: Read forward on tape ST. Read and store the words into storage locations MMMMM to (MMMMM + BBBO-1) inclusive. Variable Mode: Read forward on tape ST. Read the next block of data and store the words into sequential storage locations in ascending order beginning with MMMMM (MMMMM, MMMMM + 1, etc.). Do not store words beyond storage location MMMMM + BBBO-1. If the block is longer, an error will be indicated.
74 BBB ST MMMMM	Read Backward	Fixed Mode: Read backward on tape ST. Read and store the words into storage locations MMMMM to (MMMMM - BBBO + 1) inclusive. Variable Mode: Read backward on tape ST. Read backward the next block of data and store the words into sequential storage locations in descending order beginning with MMMMM (MMMMM, MMMMM - 1, etc.). Do not store words beyond storage location MMMMM - BBBO + 1. If the block is longer, an error will be indicated.
75 BBB ST MMMMM	Write Density 200	Write at a density of 200 pulses per inch. Fixed Mode: Write BBBO words from storage locations MMMMM to (MMMMM + BBBO - 1) inclusive. Insert a space between blocks after each group of WWWO words in accordance with summary order 63. Variable Mode: Write one block of BBBO words from storage locations MMMMM to (MMMMM + BBBO - 1) inclusive.
77 BBB ST MMMMM	Write Density 100	Write at a density of 100 pulses per inch. Note: This summary order is the same as summary order 75, except that the data is written on tape at a density of 100 rather than 200 pulses per inch.

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
	Magnetic Drum Summary Orders	
8 SS nn DD MMMMM	Read	Read nn 100-word sectors continuously from drum DD starting with sector SS. Transfer the words to sequential storage locations in ascending order beginning with storage location MMMMM. Note: To transfer a full band of data in the minimum time SS = 00 and nn = 25.
9 SS nn DD MMMMM	Write	Write nn 100-word sectors continuously on drum DD starting with sector SS. Transfer the words from sequential storage locations in ascending order beginning with storage location MMMMM. Note: To transfer a full band of data in the minimum time SS = 0 and nn = 25.
93 000 DD SB x HB	Drum Format	Interpret all future read-write head positioning summary orders for drum DD as though drum DD contained 2 X HB bands starting with band SB. Position the read-write head assembly over band SB (new 00 band). Note: There are 100 bands on a drum. The bands are normally numbered from 00 to 49 in the shift-high position and 99 to 50 in the shift-low position. (refer to section 5.3) This summary order enables the computer programmer to address the drum as though it contained fewer bands. SB = starting band (SB<50). HB = one-half the number of bands the computer programmer wishes to use on the drum at the present time (SB + HB≤50). If the complete drum is to be used SB will equal 00 and HB will equal 50. All summary orders for drum DD which follow this summary order should consider SB as equal to 00. At the completion of this summary order the head assembly will be positioned over band SB (the new 00 band).
94 xxx DD xxxxx	Next Lower Band	Position the read-write head assembly of drum DD over the next lower band. If the head assembly is now over band 00, position it over the highest numbered band in accordance with the format specified by the most recently executed 93 summary order for drum DD.
95 xxx DD xxxxx	Next Higher Band	Position the read-write head assembly of drum DD over the next higher band. If the head assembly is now over the highest numbered band

Table C-1. Summary Orders (cont.)

Format	Name	Interpretation by Processor Control Program
		as specified by the last 93 summary order for drum DD, position the head assembly over band 00.
96 x BB DD xxxxx	Position Head	Position the head assembly of drum DD over band DD in accordance with the format specified by the most recently executed 93 summary order for drum DD.
98 xxx DD xxxxx	Interlock	Do not execute any future write summary orders for drum DD.
99 xxx DD xxxxx	Remove Interlock	Execute any future write summary orders for drum DD.

APPENDIX D

ARITHMETIC AND RELATED PROCESSOR INSTRUCTIONS

D-1 GENERAL

This appendix contains a description of the more general-purpose processor instructions, including the arithmetic instructions and related instructions such as shift, comparison, and extract instructions. These instructions are used in the processor input-output control program to carry out editing, interpreting, and supervisory functions. They may also be used in a secondary or side program run on the processor. Most of the processor instructions, however, are used to communicate with and control the synchronizers, the dispatcher, the input-output devices, and the error circuits. Normally these instructions will be used only in the processor input-output control program and error routines which need not change for every program run on the computer. Since a description of these instructions to be meaningful requires a somewhat detailed knowledge of the characteristics of the devices they are used to control and communicate with, they are not included in this appendix.

D-2 INSTRUCTION FORMAT

A processor instruction word consists of 12 decimal digits, as follows:

I I N N N N N M M M M M

The meanings of the characters that are used to represent the different parts of the instruction word are:

I - instruction designator. I specifies the operations to be executed by an instruction. An instruction designator that is not in the processor repertory of instructions will cause a transfer of control to an error routine.

N - specifies the storage address of an operand for single-operand-address instructions or the address of the first operand for two-operand-address instructions. For other instructions, digits of N are used to specify the number of shifts, or the address of a synchronizer, drum, input-output device, flip-flop, or display register.

M - specifies the storage address for the result of an instruction, a second operand address for two-operand-address instructions, or a transfer-of-control address for transfer-of-control instructions.

D-3 INSTRUCTIONS

D-3.1 Conventions

The following conventions are used in presenting the processor instructions in table D-1.

rP1 and rP2 - are abbreviations for register P1 and register P2, respectively. rP1 and rP2 are arithmetic shift-registers in the processor that are used for the temporary storage of operands. When an arithmetic instruction is executed operands are shifted from rP1 and rP2 into and through an adder-comparator a digit at a time. The result is shifted into rP1 which also serves as an accumulator. The bits and digits of a word are transferred between rP1 or rP2 and the main storage completely in parallel.

C - denotes a control counter which can be assumed to contain the storage address of the instruction currently being executed.

( ) - parentheses around a symbol denotes the contents of the register, control counter, or storage location indicated by the symbol.

(C)+1-> C - denotes that the present sequence of executing instructions is continued, that is, the control counter is stepped by one to give the address of the next instruction in sequence.

M-->C - denotes that control is transferred to a new sequence of instructions starting with the instruction in storage location M.

x - lower case x denotes a digit which is not used or decoded in the instruction being described.

D-3.2 Instruction List

Arithmetic and related instructions for the processor are listed in table D-1. The format for each instruction is given together with a description of the instruction, using the conventions described in section D-3.1 above, and the execution time of the instruction, in microseconds. The execution times include the time required to obtain the instructions from storage.

TABLE D-1. ARITHMETIC AND RELATED PROCESSOR INSTRUCTIONS

	Instruction Format		Description	Time μsec.
			Data Transfer Instructions (Arithmetic)	
12	NNNNN	xxxxx	(N)-----> rP2	8
13	NNNNN	xxxxx	(rP2)---> N	8
15	NNNNN	xxxxx	(N)-----> rP1	8
16	NNNNN	xxxxx	(rP1)---> N	8
			Arithmetic Instructions	
			All additions and subtractions are algebraic Recomplementing may or may not be performed depending upon the relative magnitudes of the operands.	
09	xxxxx	MMMMM	Add-Send	
			(rP1) + (rP2)---> rP1 and (rP1)---> M	
			(rP2) is unchanged	
			If recomplementing is not necessary	12
			If recomplementing is necessary (unlike signs and |rP2|> |rP1|).	20
10	NNNNN	xxxxx	Bring-Add	
			(N)---> rP2 and (rP1) + (rP2)---> rP1	
			If recomplementing is not necessary.	12
			If recomplementing is necessary (unlike signs and |rP2|> |rP1|).	20
11	NNNNN	xxxxx	Subtract	
			(N)---> rP2 and (rP1) - (rP2)---> rP1	
			If recomplementing is not necessary.	12
			If recomplementing is necessary. (like signs and |rP2|>|rP1|).	20

TABLE D-1. (continued)

Instruction Format	Description	Time μsec.
17 NNNNN MMMMM	Bring-Add-Send	
	(N)---> rP1 and (rP1) + (rP2)---> rP1 and (rP1)---> M	
	(rP2 is unchanged.	
	If recomplementing is not necessary	16
	If recomplementing is necessary (unlike signs and |rP2|> |rP1|).	24
	Conditional Transfer of Control Instructions	
18 NNNNN MMMMM	Equality Test	
	(N)---> rP2	
	M---> C if (rP1) = (rP2)	20
	(C)+1---> C if (rP1 ≠ (rP2)	12
	(rP1) is unchanged and (N) remains in rP2.	
	All 12 digits are compared.	
19 NNNNN MMMMM	Magnitude Test	
	(N)---> rP2	
	M---> C if (rP1) > (rP2)	20
	(C)+1---> C if (rP1) ≤ (rP2)	12
	(rP1) is unchanged and (N) remains in rP2.	
	All 12 digits are compared.	

TABLE D-1 (continued)

Instruction Format	Description	Time μsec.
	Unconditional Transfer of Control Instructions	
05 xxxxx MMMMM	Transfer Control M ---> C	8
14 NNNNN MMMMM	Return Jump 050000 (C)+2---> N and M---> C	12
	Circular Shift Instructions	
07 nnxxx xxxxx	Single Precision Shift Circular (end-around) right shift (rP1) nn places. The entire word, including the sign, is shifted. nn may be any number from 00 to 29. If nn is greater than 29, control is transferred to an error routine. If nn is: 0-3 4-10 11-18 19-26 27-29	 4 8 12 16 20
08 nnxxx xxxxx	Double-Precision Shift Circular right shift (rP1)---> rP2 and (rP2)---> rP1 nn places. Both words, including the signs, are shifted. The execution times for this instruction are the same as for instruction 07.	

TABLE D-1. (contined)

Instruction Format	Description	Time μsec.
	Storage Interlock Instructions	
21 NNNNN xxxxx	Set a write interlock in the 2500-word storage unit containing storage location N so that computer number 1 cannot change the contents of the storage unit.	4
22 NNNNN xxxxx	Set a write interlock in the 2500-word storage unit containing storage location N so that computer number 2 cannot change the contents of the storage unit.	4
23 NNNNN xxxxx	Reset the write interlocks in the 2500-word storage unit containing storage location N so that computer number 1 or number 2 can change the contents of the storage unit.	4
	Display Register Instructions	
01 xxDxx MMMMM	<u>Transfer to Display Register</u> If the interlock flip-flop of display register D is not set (rP1)--> D.	8
	If the interlock flip-flop of display register D is set M--> C	12
	D is the address of a dispaly register, as follows: D=1 - 5 digit display register of computer 1 D=2 - 12 digit display register of computer 1 D=3 - 5 digit display register of computer 2 D=4 - 12 digit display register of computer 2 The contents of the five least significant digit locations of rP1 is transferred to a 5-digit display register.	

TABLE D-1. (continued)

Instruction Format	Description	Time μsec.
02 xxDxx xxxxx	Transfer from Display Register (D)---> rP1 and reset connect and interlock flip-flops of display register D. Refer to instruction 01 for the mean- of D. The contents of a 5-digit display register are transferred to the five least significant digit positions of rP1.	8
03 xxDxx MMMMM	Display Register Interlock Test M---> C if the interlock flip-flop of display register D is set.	8
	(C)+1---> C if the interlock flip-flop of display register D is reset.	4
	Refer to instruction 01 for the meaning of D.	
	Miscellaneous Instructions	
00 xxxxx xxxxx	Skip	
	(C)+1---> C	4
20 NNNNN MMMMM	Extract Send	
	Characters from digit positions of N replace characters in corresponding digit positions of rP1 if the corresponding digit poisitions of rP2 contain even characters, that is, +, 0, 2, 4, 6, 8, ., or \. (rP1)---> M.	12
	Results are retained in rP1 and M. rP2 is unchanged.	

TABLE D-1. (continued)

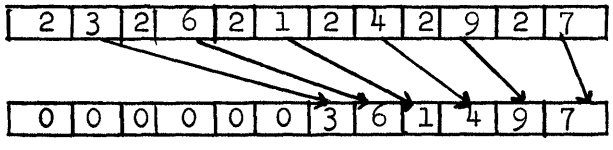
Instruction Format	Description	Time μsec.
24 NNNNN xxxxx	Alphanumeric Translate (N)--→ rP2 and the digits in every other digit position of rP2, beginning with the least significant digit position are transferred to the six least significant digit positions of rP1. For example, the word 232621242927 in rP2 would be transferred as follows:  This instruction is used to translate a word, which represents numeric data, from the LARC 2-digit alphanumeric code to the LARC 1-digit numeric code. (N) remains in rP2 unchanged.	12
25 NNNNN MMMMM	Pattern Inclusion Test (N)--→ rP2 M--→ C if for every 1-bit in the quinary portion of each digit in rP1 there is a corresponding 1-bit in rP2. The other bits, except for the check bits, are ignored.	
96 xxCxx MMMMM	Reset Disclosure Flip-Flop or Set Processor Contingency Flip-Flop. If C=1, reset the disclosure flip-flop in computer number 1. If the flip-flop fails to reset, M--→C If the flip-flop resets, (C)+1--→C If C=2, (same as for C=1, except the disclosure flip-flop in computer number 2 is reset.)	12 4

TABLE D-1 (continued)

Instruction Format	Description	Time μsec.
	If C=3, set the processor contingency flip-flop in computer number 1, thereby causing the computer to automatically transfer control.	
	If the flip-flop fails to set, M---> C	12
	If the flip-flop sets, (C)+1---> C	4
	If C=4, (same as for C=3 except the processor contingency flip-flop in the computer number 2 is set).	
97 xxCxx MMMMM	Test Disclosure Flip-Flop or Test Processor-Contingency Flip-Flop	
	If C=1, test the disclosure flip-flop in computer number 1.	
	If the flip-flop is in the reset state (C)+1---> C	4
	If the flip-flop is in the set state M---> C	12
	The disclosure flip-flop is set by the computer program to reveal the presence of a disclosure word in a predetermined storage location.	
	If C=2, (same as for C=1, except the disclosure flip-flop in computer number 2 is tested).	
	If C=3, test the processor contingency flip-flop in computer number 1.	
	If flip-flop is in the set state, (C)+1---> C	4
	If flip-flop is in the reset state, M---> C	12
	When the flip-flop is already in the set state the processor program cannot indicate a contingency to the computer program.	
	If C=4, (same as for C=3, except the processor contingency flip-flop in computer number 2 is tested).	

TABLE D-1 (continued)

Instruction Format	Description	Time μsec.
99 xxSxx MMMMM	Master Input-Output (Priority) Tests	
	Test the condition specified by S.	
	If the condition specified by S is true M---> C	12
	If the condition specified by S is not true (C)+1 ---> C	4
	S - Condition, M---> C if True	
	0 Any condition specified by S=1 through 9 is true (master-master test).	
	1 - A drum synchronizer requires attention (master sector change test).	
	2 - A tape synchronizer requires attention (master 10-word test).	
	3 - A line printer synchronizer, the electronic page printer synchronizer, or the real-time clock requires attention (miscellaneous master test I for relatively fast devices).	
	4 - A console device (console printer or manual intervention flip-flop) requires attention (miscellaneous master test II).	
	5 - A drum (1-24) requires attention (master head assembly motion test for drums 1 through 25).	
	6 - A drum (7-12) requires attention (drum head assembly motion test for drums 7 through 12).	
	7 - A drum (13-18) requires attention (drum head assembly motion test for drums 13 through 18).	
	8 - A drum (19-24) requires attention (drum head assembly motion test for drums 19 through 24).	
	9 - A tape synchronizer has completed selection.	