Abstracts of Current Computer Literature

These abstracts and the associated subject and author indexes were prepared on a commercial basis under the direction of Dr. Geoffrey Knight, Jr., Consultant to Cambridge Communications Corp., 1612 K Street, Washington, D. C., who publish Information Processing Journal, Solid State Abstracts Journal, Electronics Abstracts Journal, Cumulative Computer Abstracts, Cumulative Solid State Abstracts, and Cumulative Electronics Abstracts.

CONTENTS
ABSTRACTS .......................................................... Pages 1101-1114

Abstract Numbers
0) GENERAL; STANDARDS; EDUCATION ........................................... 6126-6127
1) LOGIC AND SWITCHING THEORY; SEQUENTIAL MACHINES .................... 6128-6140
2) DIGITAL COMPUTERS AND SYSTEMS ........................................... 6141-6152
3) LOGIC DEVICES AND CIRCUITS (HARDWARE) .................................. 6153-6155
4) DIGITAL STORAGE AND INPUT-OUTPUT EQUIPMENT .................................. 6156-6178
5) PROGRAMMING AND CODING OF DIGITAL MACHINES .......................... 6179-6207
6) LINGUISTICS, DOCUMENTATION, AND HUMANITIES .......................... 6208-6211
7) BEHAVIORAL SCIENCE, PATTERN RECOGNITION, AND ARTIFICIAL INTELLIGENCE .............. 6212-6215
8) MATHEMATICS ............................................................................. 6216-6223
9) PROBABILITY, MATHEMATICAL PROGRAMMING, DIGITAL SIMULATION, INFORMATION THEORY, AND COMMUNICATION SYSTEMS ................. 6224-6227
10) SCIENCE, ENGINEERING, AND MEDICINE ..................................... 6228-6230
11) ANALOG AND HYBRID COMPUTERS ............................................. 6231
12) REAL-TIME SYSTEMS AND AUTOMATIC CONTROL; INDUSTRIAL APPLICATIONS .......... 6232-6233
13) GOVERNMENT, MILITARY, AND TRANSPORTATION APPLICATIONS ............... 6234
14) BUSINESS APPLICATIONS OF INFORMATION PROCESSING ...................... 6235-6236

SUBJECT INDEX .......................................................... Page 1115
AUTHOR INDEX .......................................................... Page 1120
0) GENERAL; STANDARDS; EDUCATION


Exponential curves were fitted to time series data which depict the growth of computational capability, as measured by number of machines and computing power installed and on order. The latter series, for power, shows trends in the area of 15 percent and 25 percent for installations and orders, respectively. The statistical results are discussed in the context of market dynamics in the industry. The problem of lags between the order and delivery of computing facilities was then examined. Two models, developed for this purpose, regressed lagged-order variables on installations. A delivery lag of 3 to 4 quarters was found for machine installations; for power, indicative of the demand for larger computers, a lag of 4 to 6 quarters was found. Conclusions drawn relate to the problems encountered by the firm waiting for ordered computer facilities in the face of rapid technological change of computer hardware.

Methodology for Evaluating Time-Shared Computer System Usage—see 6203.

Methodology for Calculating and Optimizing Real-Time System Performance and Cost—see 6232.

Cost-Effectiveness Analysis in EDP System Selection—see 6235.

Measures, Models and Measurements for Time-Shared Information Processing Utilities—see 6207.

Public Planning Information System and the Computer Utility—see 6234.


The results of a survey of the course work done by master's degree candidates at 25 U. S. universities are presented, and some general comments concerning the emphasis of these programs are given.

1) LOGIC AND SWITCHING THEORY; SEQUENTIAL MACHINES


Decimal arithmetic, ternary encoding of cubes, and topological considerations are used in an algorithm to obtain the extremals and prime implicants of Boolean functions. The algorithm, which has been programmed in the FORTRAN language, generally requires less memory than other minimization procedures, and treats don't-care terms in an efficient manner.


The problem of solving sequential Boolean equations is shown to be equivalent to the problem of finding whether there exists a path on a labeled graph for every sequence of labels. Algorithms are given for testing whether a solution exists, and if a solution with a finite delay exists. In case of existence of solutions the algorithms provide them.


A graphical method is described that is generally useful for solving a pair of simultaneous Boolean equations, particularly the system application equation and memory characteristic equation for determining the memory element flip-flop input equations. The decision mechanism utilized for determining the Karnaugh map structure from which an optimum solution may be obtained is based totally on the behavioral characteristics of the memory element as derived from a truth table. Simple illustrative examples are included for the R-S flip-flop (with variations in dominance), the J-K flip-flop and the R-S-T flip-flop encompassing various classes of don't-care conditions.


Ternary logic circuits may be used to build digital systems. Information carried by each signal wire in such a system is greater than in a binary system. Complementary transistor configurations may be used as ternary logic circuits without resorting to complicated techniques. Design of a ternary system is accomplished by a ternary switching theory based on a switching algebra whose basic operations correspond to the ternary logic circuits. Such an algebra, and consequently the required behavior of the ternary logic circuits, is selected from the class of all possible ternary algebras on the basis of a set of theorems that aid in the formation, manipulation, and simplification of algebraic expressions. Variations of conventional simplification techniques may be applied to the resulting ternary algebra to form a combinational switching theory.


In studies of cellular logic, much work has been done with Maitra cascades. This is due to the fact that these cascades are useful as "elementary" modules in more sophisticated cellular arrays. In this paper, the application of Karnaugh maps for the Maitra realizability of any function with any number of variables is studied. A set of necessary and sufficient conditions is given to test the Maitra realizability by using only Karnaugh maps. The practical usefulness of the rules given is limited only by the difficulty of studying Karnaugh maps with more than 7 variables. The results given in this paper can also be advantageously applied to the study of rectangular arrays.


A new class of modular networks, called "modular tree networks," is presented. In these circuits every input of the first-level gate is connected to the output of a different second-level unit, and, in general, every input of every rth-level unit is connected to the output of a different (r+1)th-level unit. The networks of this type are of interest if they are "universal," that is, if a certain structure can implement any assigned function in a given number of variables. A class of universal networks is that of "decreasing- variable networks of the pth order," which are those networks in which the output signal of every second-level unit contains p variables less than the assigned function, and, in general, the signals of the (r+1)-th level contain p variables less than the signals of the rth level.


It is shown that if the family of languages accepted by a closed class of two-way balloon automata is closed under length-preserving homomorphism, then this family is an AFL closed under intersection and e-free substitution. It is then proved that the family of languages accepted by the closed class of two-way balloon automata of (non-erasing) (deterministic) stack acceptors is such a family.

Languages, Automata and Classes of Chain Encoded Patterns—see 6213.


It is shown that the equivalence problem for Lambda-free nondeterministic generalized machines is unsolvable and that this result implies the unsolvability of the equality problem for e-finite languages.

It is shown that a short proof of the equivalence of star-free and group-free regular languages is possible if one is willing to appeal to the Krohn–Rhodes machine decomposition theorem.


This paper treats the problem of finding efficient secondary assignments for synchronous sequential circuits. It presents a straightforward approach which identifies codes resulting in the least number of memory elements and sets of simple next-state functions, expressed in the sum-of-product form. A realistic cost measure is established and an algorithm is then presented. The algorithm reduces the number of choices that have to be considered and also provides simple evaluation processes to select among them. It has been applied to many state tables and is found to be much less laborious to execute, the results yielded comparing favorably with those obtained through other means. Three illustrative examples are presented.

Description of Noise Processes in Sequential Networks—see 6224.


This paper considers the problem of determining whether a sequential machine, given by its flow table, can be realized in the form of a single binary shift register. One-to-one and many-to-one assignments are considered. Binary partitions are introduced, and shift registers are characterized in terms of binary partitions. An algorithm is given for determining the required partitions for a shift-register assignment. Binary set systems are introduced and shift registers are characterized in terms of binary set systems. It is proven that there exists a sequential machine that cannot be realized by a single binary shift register of finite length. Methods for determining the set system necessary for a single shift-register assignment are given. For machines where every state has two distinct next states, a method is presented whereby a number of machines can be easily eliminated as being not realizable. It is then shown how this can be extended to all machines.

2) DIGITAL COMPUTERS AND SYSTEMS


A medium- to large-scale fourth-generation computer organization is described. An executive, a user, a user specification, a user instruction, a user instruction set, a user instruction set equivalent, a user instruction set equivalent third-generation instructions per second is predicted with the anticipated hardware capabilities and costs. The instruction stream controls the arithmetic unit and transmits commands to the data channel control unit. The data channel control unit consists of special-purpose computers for fetching, storing, address incrementing, and counting for data transfers in the buffered data channels. Instruction and command execution proceed asynchronously. An interlock occurs if an operand from a data channel is not available when needed. Concurrency is achieved without complex lookahead hardware. For time-sharing and multiprogramming, a stream of programs and their data flow off a high transfer rate disk into main memory, are executed while in main memory, and flow out to the disk.


Computer evolution and the current software situation are reviewed briefly in this article. Major topics include design approach, significance of family planning, and functional organization for development of fourth-generation systems. Use of read-only memories, associative memories, and programmable logic arrays to replace operating system and control programs and to establish logical system organization is proposed and analyzed. Hardware implementation of functions currently performed by software is advocated; specific examples are provided. Some of the characteristics of fourth-generation computer systems are indicated. Integration of these characteristics within a total system is described.

Programming Systems for Fourth-Generation Computers with Changeable Control Memories—see 6197.


The concept of a programmable word length computer has evolved though an attempt to minimize wasted storage in any form of word length control. When the computer contains 6 bits per word or 72 bits per word, storage is inevitably wasted when it is necessary to use a full word for a simple on-off switch, or to have 12-digit capacity used for 2-digit significance. Programmers have long recognized this. Thus, word packing and unpacking soon becomes a part of every programmer's repertoire. This treatise proposes a computer whose word length is placed under program control. Such a computer allows minimum wasted storage without requiring the pack and unpack routines which require substantial execution time and additional storage. It would allow a programmer who wished, for example, a 22-bit word length to set that length by an instruction. The word length could then be changed to a different bit length at any subsequent time with this set word length instruction. Several other areas of computer organization are also treated. In particular, instructions are available according to the needs of an installation rather than to the maximum size configuration possible. Communications capabilities with an external environment are extended as are internal communications for time sharing and multiprocessor.

Machine Organization for Multiprogramming—see 6201.


Algorithms are presented for multiplication and division of unsigned integer operands in which the digits normally reserved for signs participate as significant arithmetic digits with positive weight.

Arithmetical Operations over Integers with Classes and their Circuit Realization, Y. P. Sobornykov; Rept. FTD-HT-23-S80-67, 21
The paper considers a method of performing arithmetical operations of numbers with arbitrary signs in a nonexcessive, nonpositional number system of residue classes (SZK), as well as the structure of an arithmetical device realizing operations are typical of SZK and problems of linear algebra.

Arithmetic Unit for Computing Discrete Fourier Transforms of Audio Signals see 6227.


Presented here is a design for a binary adder-checker system which employs residue codes and detects any error resulting from a single fault. In an adder, special functional relationships must exist, regardless of the particular logical realization. Consequently, for adders with either serial or parallel carry propagation, the worst possible error can be described precisely. Certain residue codes may then be used to detect that error by means of a simple checking algorithm with a minimum of extra circuitry.


This note describes several methods of performing fast, efficient, binary-to-decimal conversion. With a modest amount of circuitry, an order of magnitude speed improvement is obtained. This achievement offers a unique advantage to general-purpose computers requiring special hardware to translate between binary and decimal numbering systems.


The implementation of the component division and placement module of a design automation system is described. The module accepts as input the equivalent of a logic diagram with designer specification of a class of possible logic devices to which each logic element on the diagram belongs. One must also specify the physical characteristics of the electronic devices, the number and type of each device, and the packaging system to be used. The devices on the logic diagram are then assigned to the physical hardware available. Additional designer specifications such as "logical grouping" have been implemented, and the characterization of board assignment has been generalized.


A method for computing the reliability of triply redundant majority-voted systems is proposed. Investigation indicates at least a 1000-fold increase in speed over other methods with little, if any, sacrifice in accuracy. The vehicle for studying the proposed technique was a generalized yet simple logic network. Studies also were two other approximations that are currently or were previously in use. An objective comparison of the three methods with the actual network reliability, laboriously computed, led to the above stated conclusion.


There are at least three widely used techniques for simulating faults of digital systems: manual simulation, physical simulation, and digital simulation. The purpose of this paper is to discuss the merits and drawbacks of these techniques.


Today's complex operating and computing systems make systems testing a difficult task. Major problems arise when one attempts to measure the performance of a system in a multiprogram environment and to evaluate the interfaces between computer elements, programs, and operator. Historically, testing devices were first developed to monitor system activity and to produce test data. Separate computer systems were next used to permit on-line data reduction and generation of appropriate test conditions. The purpose of this paper is to describe a hierarchical control program design which incorporates the major capabilities of the previous solutions without requiring a separate computer. This low-cost flexible technique has been applied in the measurement of various performance characteristics, in generating simulated error conditions, and in simulating machine devices and features.


This paper describes an approach to the design of very high availability digital systems. It proposes automatic error detection, self-diagnosis, self-repair, and automatic program recovery as an alternative to the classical high-redundancy techniques. Automatic error detection is performed by redundancy, using coding and duplicate or additional computation. Self-diagnosis is performed by partitioning the system into parts that can diagnose one another. Self-repair is performed by reorganizations using a control store.


The report presents final results of a third project covering research on cellular logic techniques. The objective of the research has been to develop techniques for the efficient realization of general logical functions in cellular arrays. The report covers the subjects of cellular cascades and related networks, studies of universal log modules, programmed-logic arrays, diagnosis and testing of cellular arrays, sampled-sequence detectors, serial universal logic modules, and iterative multirail cascades.


A class of networks is described that has the capability of permuting in an arbitrary manner a set of $n$ digital input lines onto a set of $n$ digital output lines. The circuitry of the networks is arranged in cellular form, i.e., in a two-dimensional iterative pattern with mainly local intercell connections.
where the basic cell behaves as a reversing switch with a single memory flip-flop. Various network forms are described, differing in the number of cells needed, in the shape of the array, and in the length and regularity of interconnections. Also discussed are some ways of setting up the array to achieve a desired permutation.


A universal logic circuit (ULC) is a circuit which can realize any logic function as a fixed number of variables by simply varying its input terminal connections. In this paper, the design of ULCs using integrated-circuit packages is discussed in detail. Various effects, such as the number of pins, the number of logic gates, and the number of logic levels in a package, on the design of ULCs are considered. A modular realization technique which uses ULCs of a small number of variables as modules to implement ULCs of various ULMs is presented. Such modules are called universal logic modules (ULMs). A diagnostic procedure for locating a subset of ULMs containing all the malfunctioning ULMs in a faulty ULC and a minimal fault-detection test for a ULM are given. A method for improving the reliability of a ULC using error-correcting codes is demonstrated.

Single Shift-Register Realizations for Sequential Machines—see 6140.

4) DIGITAL STORAGE AND INPUT-OUTPUT EQUIPMENT

Graphical Method of Solving Storage Element Flip-Flop Input Equations—see 6130.

Properties of Cellular Arrays for Logic and Storage—see 6153.

Programming Systems for Fourth-Generation Computers with Changeable Control Memories—see 6197.


This paper introduces a few coding schemes to correct bit errors in random-access memory systems, caused by defective memory cells. Cell defects within a memory could be due to fabrication errors, or due to random fluctuations in the READ-WRITE electronics associated with the memory. In the former case the defects will be permanent, and cell defects occurring (say up to 7 or 10 per word of 100 bits in the integrated memory technology). Also, it might be possible to identify such permanent cell defects before final assembly of the memory. In the latter case the bit errors will be transient, and are likely to be randomly few (1 or 2 per word of 100 bits). However, they will be randomly distributed across the length of a word being read out. In both cases, the correction technique should satisfy the following requirements: it should have small processing delay, simple parallel instrumentation, flexibility to exploit a priori knowledge on error distribution, and some protection against errors in encoding and decoding nets.

Ordered Retrieval from Associative Memories—see 6184.


In the area of magnetic associative memories the schemes so far proposed all have a low per-bit mismatch-to-match signal ratio which is one of the principal factors in determining the ultimate size (or speed) of the memory. With appropriate compensation techniques it is possible not only to make the per-bit mismatch-to-match signal ratio independent of the signal-to-noise ratio of the magnetic device, but also to provide a means of detecting the neighboring codes of a given code. Thus, both the memory size and reliability are increased. This has been confirmed both experimentally and mathematically. The memory achieves equality, similarity, and proximity searches at high speed. A fairly complete bibliography on associative systems is also included.


Laminated ferrite memories are a development of RCA Laboratories. The fabrication technique for laminates is briefly described and the results of pilot plant operation are presented. System characteristics for operation over a cycle time and capabilities suitable for computers are reviewed and evaluated with respect to other computer technologies.


Results are presented of an exploratory investigation of a large-capacity, random-access, magnetic memory combining the large-scale integration of arrays of MOS transistors and the ferromagnetic laminated-ferrite memory planes. The ferrite used has the composition of 0.38MgO-0.19MnO-0.52ZnO-0.38Fe2O3. The laminated-ferrite planes were fabricated by "doctor blading" and embossing. To circumvent difficulties in stacking because of the fabrication of the ferrite sheet (≈17 percent) during firing, pressure sintering at 1000°C and 4000 psi (typical values) has been used. A memory cross section consisting of a ferrite array with 256×100 crossovers and a word driver strip with 64 MOS transistors was assembled. On the basis of experimental tests it is judged that operation at 1.5 to 3.5-μs cycle time with 2 crossovers per bit and 2048 to 8192 words per sense amplifier/digit circuit is feasible. Extrapolation of the results to large-capacity memories with 109 bits per module is considered. Some of the difficulties in packaging and in yield are discussed.


The fast core-mass memory described in this paper has a 2D organization of the selection circuits. This organization has been chosen because of its many advantages, among which the low core-stringing cost and low dissipation within the core stack are the most noticeable. Unlike other 2D mass memories it employs a separate wire for sensing the interrogated bits. A proper termination of this sense wire allows of using only one preamplifier for 65 000 cores. In the design of the core stack a new method has been used to interconnect the basic core planes which measure only 512 by 256 bits.


This paper describes a commercial 2D 600-ns magnetic core memory system in which all major electronic functions are implemented by monolithic integrated circuits. The achievement of memory line driving substrate with logic elements on conventionally sized IC dice represents a technological first. Another unique electrical design feature is the replication of all applicable portions of the data plane peculiar to 2D organizations within each relevant geometrical plane, thus achieving the controlled signal paths necessary for high performance. This system also incorporates several unique mechanical features, the foremost of which is the complete absence of conventional backplane wiring. Another novel feature is the provision for mechanical stacking of core plane assemblies without soldered interconnections.

6162 Heating-up of Ferrite Memory Cores During Operation [in German], W. Maiwald (Siemens, Munich); Elektron. Rechenanl., vol. 10, pp. 67-72, April 1968.

In a memory core matrix a large increase of temperature may occur during operation. This heating-up is produced by the hysteresis losses in the cores and by electrical losses in the drive wires, especially in the inhibit wires of a 3D memory. The temperature rise in a core is in general dependent on the core size, the remanence induction, the drive currents, the cycle frequency, the computer program, the environmental medium (air, vacuum, embedding material), the ambient temperature, the number, the arrangement and the diameter of the wires, the size of the matrix, and the thermal contact with the ambient. These factors yield the "worst case" which cannot be valid simultaneously for all cores of a matrix. Therefore it is im-
This program is devoted to the preparation and investigation of two novel kinds of electron-beam-addressable storage elements of submicron size and densely packed arrays of these elements; also to the construction of a large-capacity, high-speed, electron-beam-addressable, data-storage system utilizing regular arrays of these elements. Work on storage mosaics was devoted to further development of techniques for the preparation of regular arrays of densely packed microcap elements including: investigations of other resists besides tetraakistriphenyl siloxitiyanium, such as KPR, Shipley's AZ 111, and poly(methyl methacrylate); argon sputtering of molybdenum films; and lead-fluoride etching of aluminium-oxide films. Appropriate combinations of these techniques appear promising. The development of a computer program for the design of electron-optical systems capable of scanning 100 000 000 elements per field of mosaic fabrication and for element address is continuing. Concurrently, the present molybdenum-lens electron-optical system is being reassembled to include a velocity-selector/electron-multiplier read-out unit and a bake-out heater.


Men communicate with each other by natural language and with machines by computer language. Building a bridge between these forms of communication has proved difficult, possibly because of misunderstandings and lack of the kind of language. This article points out that until people succeed in storing in a computer considerable areas of human experience, they are not likely to make a computer interact usefully with natural language.

The problem of generating continuous speech sound from a text source has been broken down into the generation of segmental phonemes, the generation of prosodic features, and the employment of these in a speech synthesis by rule scheme. The generation of segmental phonemes from spelling is performed by the process of decomposing the written words into their constituent morphs and combining the corresponding morphemes to obtain the target translation. A morph-to-morpheme lexicon is used, which also provided the parts-of-speech information needed by the prosodic feature generation part of the entire process.

The Sylvania Data Tablet is a graphic input device designed to improve man's communication with computers. It consists of an electronic pen, a flat writing panel, and an electronics package. As the free-moving pen is positioned over the writing panel, coordinate information in three dimensions is obtained in digital and analog form. Interfaced with a computer, this equipment facilitates entry of hand printed alphamericics, graphic design elements, and curve tracing data. The Sylvania Data Tablet is intended as an analog device in which co-ordinate information is obtained from phase measurements performed on signals received by the pen from the writing panel. Significant features resulting from this include high resolution due to the continuous nature of analog signals, and the use of a capacitive pickup (pen) since amplitude stability is not required. In addition, the Data Tablet is transparent and an inking capability is available from the pen.

Man-Machine Information System for Storage and Manipulation of Relational Sentences—see 6210.


The DISPLAY system provides pictorial representation of data selected by the user from a large structured data base. The system is unique in its approach to man-machine interaction and also in that it operates under a general-purpose time-sharing system. DISPLAY features include: light-pen inputs from a dynamically up-


This program is devoted to the preparation and investigation of two novel kinds of electron-beam-addressable storage elements of submicron size and densely packed arrays of these elements; also to the construction of a large-capacity, high-speed, electron-beam-addressable, data-storage system utilizing regular arrays of these elements. Work on storage mosaics was devoted to further development of techniques for the preparation of regular arrays of densely packed microcap elements including: investigations of other resists besides tetraakistriphenyl siloxitiyanium, such as KPR, Shipley's AZ 111, and poly(methyl methacrylate); argon sputtering of molybdenum films; and lead-fluoride etching of aluminium-oxide films. Appropriate combinations of these techniques appear promising. The development of a computer program for the design of electron-optical systems capable of scanning 100 000 000 elements per field of mosaic fabrication and for element address is continuing. Concurrently, the present molybdenum-lens electron-optical system is being reassembled to include a velocity-selector/electron-multiplier read-out unit and a bake-out heater.


Men communicate with each other by natural language and with machines by computer language. Building a bridge between these forms of communication has proved difficult, possibly because of misunderstandings and lack of the kind of language. This article points out that until people succeed in storing in a computer considerable areas of human experience, they are not likely to make a computer interact usefully with natural language.

The problem of generating continuous speech sound from a text source has been broken down into the generation of segmental phonemes, the generation of prosodic features, and the employment of these in a speech synthesis by rule scheme. The generation of segmental phonemes from spelling is performed by the process of decomposing the written words into their constituent morphs and combining the corresponding morphemes to obtain the target translation. A morph-to-morpheme lexicon is used, which also provided the parts-of-speech information needed by the prosodic feature generation part of the entire process.

The Sylvania Data Tablet is a graphic input device designed to improve man's communication with computers. It consists of an electronic pen, a flat writing panel, and an electronics package. As the free-moving pen is positioned over the writing panel, coordinate information in three dimensions is obtained in digital and analog form. Interfaced with a computer, this equipment facilitates entry of hand printed alphamericics, graphic design elements, and curve tracing data. The Sylvania Data Tablet is intended as an analog device in which co-ordinate information is obtained from phase measurements performed on signals received by the pen from the writing panel. Significant features resulting from this include high resolution due to the continuous nature of analog signals, and the use of a capacitive pickup (pen) since amplitude stability is not required. In addition, the Data Tablet is transparent and an inking capability is available from the pen.

Man-Machine Information System for Storage and Manipulation of Relational Sentences—see 6210.


The DISPLAY system provides pictorial representation of data selected by the user from a large structured data base. The system is unique in its approach to man-machine interaction and also in that it operates under a general-purpose time-sharing system. DISPLAY features include: light-pen inputs from a dynamically up-
A multilevel graphical data structure for an 18-bit digital computer is described. Graphical entities defined by the data structure may be linked together by a ring structure, logically tied coordinates, or other independent parameters. The structure is ideally suited for use with a large, delayed-access, mass-storage device, as the displayed picture representing the graphical entities can be generated independently of their logical linking.


A special-purpose time-sharing executive program for the PDP-1 computer is described. Input-output transfers, multiplex-user operation, interuser communication, and intermodule transfers are interpreted and performed by the executive program. Although the program reflects the hardware configuration of the Dynamic Experimental Processor (DX-1), the techniques used are applicable to other similarly configured computer systems.


The programs of the DX-1 Display System are documented. Descriptions include program functions, calling and exiting sequences, common storage registers, logic of operation, and program listings.


This report discussed the design and use of equipment built for data communication to and from a PDP-8 through a 201A data set. The purpose of the data communication interface is to allow a PDP-8 to send and receive digital data through a 201A data set in half-duplex mode. Basic design objectives and decisions are described first. A brief overall system description together with a sketch of a data format scheme and programming considerations is followed by a detailed description of the interface logic.

5) PROGRAMMING AND CODING OF DIGITAL MACHINES

Impact of Fourth-Generation Software on Hardware Design—see 6142.

Programmed Word Length Computer—see 6143.

Minimization of Wasted Storage—see 6143.

Programmed-Logic Arrays—see 6153.


A program scheme can be represented by a directed graph with its vertices or arcs labeled according to some fixed scheme. Such
a graph explicitly indicates the control structure of the program scheme and is in standard form is it satisfies a given set of constraints. Two such standard forms have been investigated in this report. Part 1 of the report considers the block form and Part 2 the $K$-form, respectively. Each part is self-contained.

6180

A Note on Segmentation of Computer Programs, A. T. Berziss (U. Pittsburgh); Information and Control, vol. 12, pp. 21-23, January 1968.

It has been shown that the cycle picture of the direct form of a computer program can be used to solve the segmentation problem. A simple algorithm for finding the cycle picture is given.

6181


For the segmentation of a program it is useful to have a reasonable estimation of the values of $S_{ij}$, where $S_{ij}$ is the mean value of the number of jumps from the $i$th instruction in the run time. In the cases where the $S_{ij}$ are estimated directly, the structure of the whole program must be generally taken into account. It is very difficult for the programmer and/or the translator to obtain a good estimation of the $S_{ij}$. It is easier to estimate $n_{ij}$ but the quantities $p_{ij} = S_{ij} / (\sum_{i > j} S_{ij})$, where $c_{ij}$ is an arbitrary positive constant for each $i$. Although the $p_{ij}$ are, for each $i$, proportional to $S_{ij}$ the estimation of $p_{ij}$ is easier, because one must estimate only the “probabilities” of events where instruction $i$ is executed after instruction $j$. This estimation can often be done without considering the structure of the whole program. In the first part of the paper, using the theory of the Markov chains, an algorithm for the computation of the $S_{ij}$ from the $p_{ij}$ is found, and some ways of obtaining estimates of the $p_{ij}$ are given. In the second part a variant of this algorithm is derived, avoiding the necessity of computing involving large matrices.

6182


Two classes of issues are relevant to the expression of processing in computer programs. Detailed technical issues concerned with sufficiency and reproducibility are largely separable from the less frequently attacked issues of ease of use and other human factors. The expression of control and the processing is only a special case of the problem of "structuring" components of a program from the standpoint of control and data relationships. A set of top level language constituents is described which permits the simple, natural expression of sequence and parallelism constraints and which requires minimal additional analysis by the programmer. Parallelism is treated as an intermodular function controlled by explicit data transmission. The functions of transferring control, communicating input parameters, and communicating output values are separated, providing several distinct structures. This system unearls of the normal subroutine call. Parallelism and module activation are regulated by the presence of input data. The structures constructed by this mechanism are inherently reproducible.

6183


A second method for partial order control using the results of the previously reported parallelism detection algorithm is given. A comparison in cost of representation and use is made among the two order control methods and sequential control. Parallelism among executions of loop bodies and among nested and interleaved loops is examined. A system design for parallelism exploitation is described. This is a block oriented random access memory (BORAM) for secondary storage to minimize access time. Some characteristics of dynamic sequencing procedures for selection among ready tasks are given.

6184


An efficient method of ordered retrieval of binary words from an associative memory as described by Lewin, is based on the use of special read-out circuits which indicate the digit values present in the individual digit columns of the memory. Thus the circuits indicate whether the individual digit columns contain digits of both values, or of only one value, or contain no digits at all (i.e., that the memory is empty). The proof of these circuits, which in this paper are termed column value indicators, reduces considerably the number of memory accesses necessary to retrieve in order a number of distinct binary words from the memory. Lewin proves that, for the read-out by the described method of $m$ distinct binary words, $2m-1$ memory accesses are necessary. (Thus he proves that the number of necessary memory accesses of his method, unlike those of other methods, is independent of the word length.) In this paper a very simple proof of this theorem obtained from the same elementary aspects of the structure of sets of binary numbers is presented.

6185


A set-theoretic data structure (STDS) is virtually a "floating" or pointer-free structure allowing quicker access, less storage, and greater flexibility than fixed or rigid structures that rely heavily on internal pointers or hash-coding, such as "associative or relational structures," "list structures," "ring structures," etc. An STDS relies on set-theoretic operations to do the work usually allocated to internal pointers. The question in an STDS will be a set-theoretic expression. Each set in an STDS is completely independent of every other set, allowing modification of any set without perturbation of the structure; while fixed structures resist creation, destruction, or changes in data. An STDS is essentially a meta-structure, allowing a question to "direct" the structure or data-flow. A question establishes which sets are to be accessed and which operations are to be performed within and between these sets. In an STDS there are as many "structures" as there are combinations of set-theoretic operations; and the addition, deletion, or change of data has no effect on set-theoretic operations, hence no effect on the "dictated structures." Thus in a floating structure like an STDS the question directs the structure, instead of being subservient to it.

6186


An important consideration in the design of programming systems for the management of large files of data is the method of treating hierarchical data (that is, data among which logical relationships exist at more than two levels). Recent systems have accomplished this by simply duplicating some essential item of data at several levels. Such duplication makes the storage of even small data bases inefficient; for large masses of data, storage becomes economically unfeasible. Other systems have provided the means to specify and construct hierarchies, but none have provided language that affords control over retrieval and output levels, and control over the scope of output. Within the Time-Shared Data Management System (TDMS), currently being produced at System Development Corporation, a method has been devised for maintaining hierarchical associations within logical entries of a data base. Basically, the technique permits the automatic association of related data through a device known as a repeating group concept is. This paper describes how this technique is implemented in the language and tables of TDMS.

6187


On the design of a partial pass block sort with arbitrary range of key and number of work files is described. The design is a generalization of the Partial Pass Count Sort by Ashenhurst and the Amphiabaen Sort by Nagler. The power of the sort is tabulated for various sizes of input file and number of work files. Consideration is given to the problem of combining a block sort with internal sorts, and to the best use of direct-access storage devices.

The incompatibilities encountered in attempting to consolidate diverse files are inherent not so much in the nature of the data involved as in the nature of the input process. Substituting one input process for another changes the constraints imposed. Thus, information which is keypunched is subject to more constraints than if it were entered on forms or data sheets. The author describes the direct encoding of forms. The method facilitates both gathering the input and generalizing the data base. Input generation differs little from the filling-in of ordinary forms on an ordinary typewriter. The programs used with the encoder can detect all item according to the form on which it was entered, and according to its location thereon. New forms may be created if and when required, without involving additional programming, and their input added to the data base without necessitating its reorganization.

DATAPLUS Language for Real Time Information Retrieval from Hierarchical Data Bases — see 6211.


A class of algebraic languages is described with the following properties: 1) no side effects, 2) statements may appear in any order, 3) all common subexpressions may be combined, 4) the compiler can detect all opportunities for simultaneous execution which are independent of run-time data values; and 5) the compiler can determine allocation and release of storage with extreme efficiency.


FORMULA ALGOL is an extension of ALGOL 60 incorporating formula manipulation and list processing. This manual describes the use of the version of FORMULA ALGOL which is presently running at Carnegie-Mellon University.

6191 Features of the GIER ALGOL 4 System, P. Naur (A/S Regencentralen, Denmark); BIT, vol. 8, no. 1, pp. 36–42, 1968.

A brief survey is given of the new features included in the GIER ALGOL 4 compiler system, as compared with its predecessors. As background the more recent additions to the hardware of the machine are described. The new features aim primarily at a more effective use of the existing machine facilities, including facilities existing in some installations.


This report is a description of the preliminary specifications including calling sequences for an input-output item handling system to interface between FORTRAN programs and the 1108 EXEC 8 data handling routines. It is intended that this interface provide to a FORTRAN program all the item handling capabilities that are available to an assembly program under EXEC 8 control, with some additional programming conveniences incorporated.


The usual method of dealing with frequently encountered computational problems is to write a large special-purpose program. Generally it turns out that flexibility is achieved at the expense of considerable clerical work being required by every user and, further, that there is never quite enough flexibility, so that continual reprogramming is necessary. Suggested and discussed is the use of precompilers, which, in effect, create a special-purpose program for each particular task. Experience has shown that even users not too familiar with FORTRAN can make efficient use of such precompilers to generate special-purpose programs of their own.


LISP A is a modification and extension of LISP 1.5. Besides minor modifications, two new types of functions have been added to the language. One type (symbolic functions) is used to create and extend the data base. If IFATHER is a symbolic function, evaluation of (IFATHER JOIN DICK) will create a representation for the relation in the data base, without asserting its truth. This representation can then be used with conventional LISP functions, which set it true, ask whether it is true, etc. The other type (RHO-expressions) can be used to write a kind of rules of inference, which are automatically triggered in desired situations. The LISP A system is governed by such RHO-expression operators, which trigger each other. There is no coherent program, just a set of operators which communicate through the changes they cause in the data base. The paper gives a general description of the LISP A system.


The rapid growth of the extent of programming languages requires improved methods for language definition. This paper presents methods for the precise, complete and unambiguous definition of programming languages which have been developed as tools for a precise definition of PL/I. The definition system for PL/I consists of separate processes which allow the separation of various concepts. Specifically, a clear separation of all problems of program representation and notational conventions from the functional concepts has been achieved. An abstract syntax is defined which specifies the syntax of a normal form of PL/I programs. A translator converts concrete programs to their normal form. The definition of semantics of PL/I is given by interpreting the normal form of programs on an abstract PL/I machine. For the description of programs in their normal form as well as the state of the PL/I machine as a class of abstract objects and appropriate functions on these objects have been introduced.


The Transformational Grammar Tester (TGT) is a special-purpose interpretive system for interactive use with a time-shared computer via a teletype and a CRT option. The language accepted by the interpreter allows the user to create and name data structures in the form of trees and manipulate these tree structures by means of pattern-matching/pattern-changing rules (transformations). These transformations allow the specification of subpatterns as well as the imposition of special conditions on the tree nodes. This system is especially designed for the use of transformational linguists in building and validating grammars describing natural languages, where the trees represent sentence structures and the transformations operate cyclically starting at the lowest embedded sentence structure and working upward.


The possible introduction of changeable control memories in fourth-generation computers may color the manner in which programming systems are designed and implemented. With this type of machine, one will be able to implement varieties of instruction sets by writing microprograms to interpret them. The trade-offs involved in deciding what part of a system should be defined by the hardware and what part is to be handled by software will provide a challenge for future systems designers and implementers. The purpose of this paper is to introduce some of the considerations required in evaluating various forms of instruction streams. These instruction streams are organized for the convenient execution of translated programs that are encoded in programming languages such as FORTRAN, COBOL, ALGOL, PL/I, etc. Six functionally equivalent but
structurally different instruction stream forms are discussed as to composition and facilities required in their interpretation. Several of the instruction-stream forms require the use of push-down stacks, and a notation is provided for describing their structure and manipulation. After the various instruction-stream forms are presented, they are compared quantitatively and contrasted with respect to four attributes: conciseness, complexity, dynamic capability, and flexibility.


Queuing theory and statistical methods are used in this paper to derive formulas for determining average turnaround time in teleprocessing systems that handle messages on a priority basis. This information is needed to ensure, for example, that messages are processed within acceptable time limits and that an efficient use is made of system resources. Factors considered in arriving at the formulas described include waiting time in message queues, message processing time, I/O waiting time, and delays for higher priority processing. Results conform closely with those obtained from simulation studies.


Designers and users of multitasking operating systems must be alert to the problem of task deadlock, which prevents the affected tasks from being completed. This paper describes the conditions that can result in task deadlock in any multitasking systems. Also discussed are techniques for avoiding deadlock in both operating system and application program design. Finally, it is shown how these techniques were applied in the design of the System/360 Operating System job initiator, the part of the system that allocates major resources needed to execute jobs.


This paper describes the properties and the function of the Multiple Operating System HYDRA EL X8, developed at the University of Karlsruhe. Part I contains a description of the way in which jobs are prepared, set-up, and introduced into the system. Furthermore, the processing of these jobs and various reactions of the system, such as tasking, priorities, background work, scheduling, storage allocation, etc.

Statistics Gathering and Simulation for the Apollo Real-Time Operating System—see 6229.


This paper is intended as an introduction to some of the basic concepts of multiprogramming for readers who wish to study the more specialized literature in this field. It attempts to develop a framework for the discussion of multiprogramming which motivates the forms of machine organization used in current multiprogramming systems.

The key requirement in multiprogramming systems is that information structures be represented in a hardware-independent form until the moment of execution, rather than being converted to a hardware-dependent form at load time. This requirement leads directly to the concept of hardware-independent virtual address spaces, and to the concept of virtual processors which are linked to physical computer resources through address mapping tables. The structure of the class of hardware-independent virtual processors in the IBM/360 Model 67 and GE 645 systems is developed in some detail. Questions of efficiency of throughput in the resulting class of computer systems are considered.


Results are summarized from an empirical study directed at the measurement of program operating behavior in those multiprogramming systems in which programs are organized into fixed length pages. The data collected from the interpretive execution of a number of paged programs are used to describe the frequency of page faults, i.e., the frequency of those instants at which an executing program requires a page of data or instructions not in main (core) memory. These data are used also for the evaluation of page replacement programs and for assessing the effects on performance of changes in the amount of storage allocated to executing programs.

Testing the Performance of Systems in a Multiprogramming Environment—see 6151.


The development of time-shared computer systems has led to major technical and philosophical changes in the computer field in this decade. A large number of designers, manufacturers, and users of such systems have expended great amounts of effort in the development of the capabilities of the computer. However, very little or no effort has yet been expended to evaluate these systems in terms of their usefulness for present or future customers. The research reported has focused on the development of a methodology through which time-shared computer system usage can be evaluated. It is based on a study of the characteristics and design of present and proposed computer systems, as well as relevant behavioral theory and research. Five categories of variables are included in the resulting methodology, namely those which are measures of 1) the cost of using the system, 2) the performance produced through the use of the computer system, 3) the speed with which results could be produced, 4) the amount of learning resulting from the use of the computer system, and 5) the attitudes of the users of the computer system.


The Berkeley Time-Sharing System is divided into three major parts: the monitor, the executive, and the experiment. Only the first two of these are discussed in detail in this manual. The manual attempts to describe exhaustively all the features of the monitor and in addition to give a number of implementation details. It also describes those features of the executive which can be invoked by a program. The word monitor is used to refer to that portion of the system which is concerned with scheduling, input-output, interrupt processing, memory allocation and swapping, and the control of active programs. The executive is concerned with the control of the directory of symbolic file names and backup storage for these files, and various miscellaneous matters. Other parts of the executive handle the command language by which the user controls the system from his teletype, the identification of and specification of the limits of their access to the system. The subjects are discussed in the executive reference manual.


The Project GENIE operating system is a medium-scale multiaccess computational system which implements a powerful and complex user machine. It is the role of the command language (here called the EXECU- tive) to provide some tools to control this user machine, and to provide those services which users have come to expect of conversational systems. This document describes the system command language.

The time-sharing system involving memory relabeling, common routines, and duplex teletype operation has been in operation since April 1965. The system is highly flexible and can provide a response time of less than one second. Memory relabeling is accomplished with no increase in access time. The number of processor modes is small (two), and mode transitions are done in such a way as to enable interrupt and user-called system routines to be independent of mode. The user machine is clean and well defined. Input/output is simpler, more foolproof, and device-independent. The user is given a variety of other services ranging from generalized file-handling capability to string processing to assemblers, compilers, debuggers, and editors.


There are two major properties of an information processing utility which are forcing intense efforts towards realization despite enormous technological, legal and social difficulties. The first of these is the value gained by users from direct access to shared resources made available through the investment of a utility supplier. The second of these is the qualitative change in the value of information blanks which may increase in value as a result of use by distributed users. The former property may be realized by distributed resources enhanced by shared centralized resources. The latter property must be realized by integrated shared centralized resources whose efficacy may be enhanced by distributed resources. Neither property may be realized without the design of an effective time-shared processor system. This paper presents a characterization of the time-sharing system environment, a set of measures associated with the system, an integrated summary of models and a discussion of measurements. In the final section new proposals for systems, measures, models and measurements are considered.

On-Line Multiprocessing Interactive Computer System for Neurophysiological Investigations—see 6239.

6) LINGUISTICS, DOCUMENTATION, AND HUMANITIES

Transformational Grammar Tester—see 6196.

Languages Accepted by Two-Way-Balloon Automata—see 6134.

Languages, Automata and Classes of Chain-Encoded Patterns—see 6213.

Unsolvability of the Equality Problem for c-Finite Languages—see 6135.

Classification Scheme for Pattern Languages—see 6213.

Application of Linguistic Theory to Pattern Analysis and Recognition—see 6213.


In order to make a syntax-directed parsing processor more efficient it is useful to impose a partial ordering on the syntax elements, i.e., terminal characters and defined terms of the language. Algorithms are presented which transform the graph G into a acyclic graphic graph G' and which from G' obtain the desired ordering.


The report describes a scheme for recording text on computer-readable form in such a way that all meaningful typographical distinctions are represented in a standard way. Provision is made for texts in different languages and different alphabets and for subsidiary material such as parallel translations and comments of interest to users and librarians. The basic set of encoding conventions is indefinitely extensible to accommodate new kinds of material. Very large bodies of data require special facilities, and these have been provided by embedding the text encoding scheme in a general file maintenance system. Computer programs are described which simplify conversion of text from these various sources into the standard format. The final section discusses the problem of printing text which has been recorded in the standard format and describes a flexible program for doing this.

Man-Machine Communication by Natural Language—see 6166.


This report is written as a contribution to the design of a man-machine information system in which the storage and manipulation of "relational sentences" is a major activity. A new representation for the definition of a relation (which becomes the basis for logical deductions) is presented. It unifies the classical properties as variations of a single process, the partitioning of small directed graphs into accepted, Inferential and Contradictory categories. While the method is not adequate by itself as a basis for program design, its systematic treatment of relational properties makes it of mathematical interest. Specific modifications which appear to overcome the known inadequacies are presented and directions for future work are indicated.


DATAPLUS is a query language designed for real-time information retrieval from a hierarchical data base. It was implemented in FORTRAN on a commercially available time-shared computer system. Syntactically and semantically, DATAPLUS resembles English. Information requests are presented to the computer in the form of a continuous flow of statements written in the DATAPLUS language. Flexibility is provided for addressing and manipulating the data at various levels of the hierarchical structure. The user is given the ability to create functions of items appearing in the data base, and to operate on these functions in the same way that he operates on items already in the data base. The language is thus easy to use and provides substantial information processing capabilities.

Ordered Retrieval from Associative Memories—see 6184.

7) BEHAVIORAL SCIENCE, PATTERN RECOGNITION, AND ARTIFICIAL INTELLIGENCE


This paper attempts to lay bare the underlying ideas used in various pattern classification algorithms reported in the literature. It is shown that these algorithms can be classified according to the type of input information required and that the techniques qf estimation, decision, and optimization theory can be used to effectively derive known as well as new results.


By treating patterns as statements in a two-dimensional language, it is possible to apply linguistic theory to pattern analysis and recognition. This report presents an approach to a classification scheme for pattern languages that could provide information about types of programs and computation facilities capable of meeting particular pattern analysis and recognition requirements.
Consideration is restricted to line patterns encoded in the chain code developed by Freeman. This encoding method represents a line pattern by a sequence of octal digits called and U. Kuhl has extended to other forms of encoding when translators between codes can be built. The report compares languages formed by Boolean functions of languages and by the concatenation strings of a number of languages with pattern languages. Pattern languages based on families of equations in two variables and formed from chains of straight lines, circles, and circular arcs are related to string language classes. Pattern properties, including closure, self-intersection, convexity, and periodicity, are examined. Pattern languages are also considered that are similar in various ways to an arbitrary given chain. Although the pattern considered in this report are relatively simple ones, it appears possible by means of a language analyzer to derive a pattern recognizer that can be used to recognize and manipulate patterns containing more complex patterns.

Manipulation of Tree Structures by Pattern Matching/Pattern Changing Rules—see 6196.


An unconventional approach to character recognition is developed. The resulting system is based solely on the statistical properties of the language, therefore it can read printed text with no previous training or a priori information about the structure of the characters. The known letter-pair frequencies of the language are used to identify the printed symbols in the following manner. First, the scanned characters are partitioned into distinct groups of similar patterns by means of a distance measure. Each class (at most 26 are permitted) is assigned an arbitrary label, and an intermediate tape, containing these temporary labels of the symbols in the original sequence, is generated. In the second phase of the program, the matrix of bigram frequencies of the labels is compared to a frequency matrix obtained from a large sample of English text. The labels are then assigned alphabetic symbols in such a way that the correspondence between the two matrices is maximized. The method is tested on a 100,000-character data set comprising four markedly different fonts.


This paper describes a technique for determining points of inflexion by successive interpolation. The convergence properties of the method are also discussed.


An exchange algorithm for uniform spline approximation with prescribed knots is developed. Implementation and numerical experience are discussed, and an ALGOL program is appended.

Principles of an Interval Calculus for Matrices and Some Applications [in German], N. Apostolatos and H. Kuhl (U. Karlsruhe, Germany); Elektron. Rechenanl., vol. 10, pp. 73-83, April 1968.

This paper deals with an interval-calculus for matrices whose elements are intervals over the real number field. The construction is such that it can be transferred to other fields of the interval-analysis, e.g., a complex interval-calculus or an interval-calculus for complex matrices. The last section gives some applications of the former theory.


In the numerical integration of an improper integral of the first kind, it is customary to truncate the integral when the change yielded by the last iteration is less than some predetermined constant. The efficiency of such integration schemes can often be improved by use of recent advances in the theory of nonlinear transformations; however, for several important integrals, e.g., integrals whose integrands are rational polynomials, these transformations fail to yield much improvement. In this paper, several methods of convergence improvement are developed which greatly improve convergence of some improper integrals, including the integrals of rational polynomials.


A new method is proposed to calculate the intersection points of two plane curves. The theory of singular points of a system of two differential equations is used in developing the method. The intersection point to be determined is identified with such a singular point and appropriate modifications are applied to the system to ensure that the singular point be stable, i.e., all integrals which start in the neighborhood of the singular point will always approach this point if the integral parameter tends to infinity. Indeed, a method is presented for systematically searching for all intersection points in a prescribed rectangular area.


The use of a fourth-order Runge-Kutta method to solve a system of differential equations, estimates of the local discretization errors of the solutions are needed in order to adjust the step size. Unfortunately, the known estimating schemes must start at the third integration step if no additional derivative evaluations of the system are available, or require that such derivative evaluations be made to get estimates before the third step. An estimating scheme is presented in this paper which allows an estimate to be obtained at the second integration step, with one additional derivative evaluation of the system, and at each step thereafter without additional derivative evaluations. The estimate can also be employed as a corrector to get more accuracy in the solution. Experimental results are given in verification of the estimate.


A computer program is presented for solving a system of two quasilinear partial hyperbolic differential equations in two independent and two dependent variables by the method of characteristics. There is a discussion of the characteristic equations and the compatibility relations. The second-order finite difference approximations to these are given along with a method to solve the difference equations.

The acquisition of starting values is one of the chief difficulties encountered in computing a numerical solution of Volterra's integral equation of the second kind by a multistep method. The object of this note is to present a procedure which is derived from certain quadrature formulas and which provides these starting values, to provide a sufficient condition for the approximate solution and the error, and to give a numerical example.

Description of a Set-Theoretic Data Structure—see 6185.

Graph-Theoretical Solution of Sequential Boolean Equations—see 6129.

Transformation of Program Schemes to Standard Form Using Graph Theory—see 6179.

Segmentation of Programs Using Graph Theory—see 6180.

Partial Ordering of the Syntax Elements of a Language Using Graph Theory—see 6208.

Graph-Theoretical Treatment of Relational Properties—see 6210.

9) PROBABILITY, MATHEMATICAL PROGRAMMING, DIGITAL SIMULATION, INFORMATION THEORY, AND COMMUNICATION SYSTEMS


Recent studies have shown that many noise processes in sequential networks can be described as a multistational process, a Markov process, or a linearly dependent process. This paper extends the results of those studies to show how the statistics of any sequence selected from processes of this type can be calculated. Through the use of z-transform techniques, it is shown that the characteristic roots of the characterization matrix which describe the process are useful in categorizing the statistical properties of the process. It is also shown that the logical combination of two or more linearly dependent sequences is another linearly dependent sequence. The relationship between the characterization matrices of the input sequences to a sequential machine and the characterization matrix of the output process is derived.


This is a discussion of some on-line random number generators, requiring a single FORTRAN instruction, together with a description of some short FORTRAN programs which mix several such generators.

Evidence suggesting that the simple congruential generators are unsatisfactory continues to grow; one of the most promising alternatives is to mix several simple generators. These composite generators do better in various tests for randomness than do the simple congruential generators used at many computer centers.

Noise in a Balanced Capacitor Read-Only Store—see 6164.

Statistics Gathering and Simulation for the Apollo Real-Time Operating System—see 6229.


The existing procedures for digitally simulating nonlinear physical dynamic systems gravitate toward either extreme simplicity or excessive complexity. In the first case, only crude approximations are generally produced and in many cases the resulting simulations are, in fact, grossly in error. The second case yields excellent replications of the physical system's dynamics. However, the price which must be paid to design such simulations may be considerable. This paper presents new techniques employed for digital modeling of nonlinear continuous systems. The method is presented for obtaining real-time simulations of nonlinear systems which provide accurate results with a minimal expenditure of design effort or computation time. An adaptive filtering technique is also described in which a time-varying compensatory device is employed in conjunction with the discrete system model to increase simulation accuracy. Selected examples indicate that such easily developed and implemented simulations compare very favorably with simulations produced utilizing very extensive design procedures.

Comparison of Fault Simulation Methods for Digital Systems—see 6150.

Models of Order-Delivery Lags for Computers—see 6126.

Residue-Code Detection of Errors in Binary Adders—see 6146.

Codes for Error Correction in Random-Access Memory Systems—see 6156.


A digital processor capable of computing the discrete Fourier transform for a range of audio signals in real time has been built as part of a facility to conduct research in signal processing. The digitized sample values can be complex. The arithmetic unit is configured to perform complex connectives, and automatic array scaling is used to make numerical accuracy independent of signal level. The Cooley–Tukey "fast Fourier transform" is the algorithm used.

10) SCIENCE, ENGINEERING, AND MEDICINE


This note defines a general F matrix that arises from a tree that is formed according to a specific element priority. An alternative based on this matrix is presented to the usual method of repeated matrix manipulation for arriving at the solutions of various network quantities. The conditions required for this alternative and its limitations are also presented.

Digital Modeling and Simulation of Nonlinear Systems—see 6226.


Since each Apollo manned space flight makes new demands on the computer configuration and operating system, data-processing efficiency is tested before each flight by simulation described in this paper. Discussed is the dynamic gathering of operating system performance data during real-time simulation, achieved by incorporating appropriate routines in the Apollo control program. The data thus collected is used as input to improved system models. The effect of the statistics gathering routine on systems performance can be measured.


A community of neurophysiologists share a computer facility especially geared to their needs for A/D data collection, D/A display of data analyses, and relay driven control of experimental events. Portable console substations in the investigators' laboratories enable on-line, time-sharing access to these functions by means of console programming and public and private program execution with parameter entry via a macrolanguage appropriate to I/O computational functions and the fast-decision environment peculiar to neurophysiological research. When intermediate computations may be performed offline, a batch-processing facility with compatible I/O magnetic tape formats is available.
11) ANALOG AND HYBRID COMPUTERS

6231

It is noted that the computing capabilities of a combination of a digital computer and an analog computer connected by a communications unit considerably exceed the capabilities of the digital and analog devices taken separately. The combination is especially valuable for small computer centers which include one or two "Ural"-type limited-speed digital computers or a "Minsk" and several MN-7 type or MNB analog computers. It is reported that the Problems Laboratory of Automation and Telecommunications and the computer center of the Moscow Power Institute have developed and constructed a combined computer system based on the "Minsk-11" computer and an MNB type analog computer, using a universal connecting unit. A description of the individual segments of the system is presented.

12) REAL-TIME SYSTEMS AND AUTOMATIC CONTROL; INDUSTRIAL APPLICATIONS

6232

The continually increasing size, complexity, number of types, and cost of data processing systems are causing serious re-examination within government and industry of the criteria for and methods of calculating and optimizing data processing system cost and performance. Real-time data processing systems as typified by the automated airline reservation system are discussed in this paper. Criteria for evaluating performance are described; a methodology for calculating and optimizing is outlined; and the method is illustrated by carrying out a portion of the performance calculation and the optimization of a drum-oriented message switching system.

On-Line Multiprocessing Interactive Computers System for Neurophysiological Investigations—see 6230.

6233

Project networks are used in PERT and CPM. An algorithm is given for constructing project networks directly from the project precedence relations. The algorithm creates "dummy" activities and topologically orders the arcs and nodes. The number of nodes created is minimal for the given precedence relations. It has been experimentally programmed in FORTRAN II for the IBM 7094.

13) GOVERNMENT, MILITARY, AND TRANSPORTATION APPLICATIONS

6234

This paper attempts to anticipate the development of two types of organizations: the public planning information system and the "computer utility." Public planning information systems are computer-based systems for storing, retrieving and analyzing data in support of relatively long-range, urban-planning activities. The computer utility is an emerging form of organization providing computer services to all types of users and may in the future distribute computer services much as electric power is distributed today. Both the planning system and the computer utility involve information and the computer services which make information useful, but the planning system is an information-oriented adjunct to public planning while the computer utility is a computer-oriented service to an entire urban, regional, or national area. Despite these differences the planning system is evolving into a form similar to that of the computer utility, with the exception of the planning system's general restriction to a select governmental, semipublic area. This similarity suggests technical and economic benefits from some combination of the two, in which either the planning system becomes a public computer utility or the computer utility provides services to, or supplants, the planning system.

14) BUSINESS APPLICATIONS OF INFORMATION PROCESSING

6235
Application of Cost-Effectiveness Analysis to EDP System Selection, J. D. Porter and B. H. Rudwick (Mitre, Bedford); Rept. MTR-527, 60 pp., March 1968; U. S. Gov't R & D Repts., vol. 68, p. 68(A), June 10, 1968. AD 667 522

A conceptual approach for evaluating and selecting among alternative Electronic Processing (EDP) systems proposed to meet a set of EDP user needs has been developed by applying cost-effectiveness methods and techniques to the source selection problem. The report provides a framework that allows the EDP system evaluator to combine the selected relevant system performance measures and the related cost elements to arrive at a rational defendable selection decision.

6236

In the spring of 1965, the Bell Telephone Company of Pennsylvania undertook a trial designed to eliminate most of the paper records used for negotiations in a business office. The objectives were to computerize these files, recall the records in real-time with video display devices, and direct the customers' incoming calls with an Automatic Call Distributor. On August 28, 1967, a Service Representative successfully handled the first customer contact. Currently, an average of 5000 contacts weekly are handled at twenty display terminal positions. The paper discusses the evolution of the trial, describes hardware selection, pioneering software solutions, and new testing techniques associated with the mechanization of approximately 250,000 constantly changing records comprising 88,000 customer accounts. It points up the problems associated with the development of a system combining complex applications functions, a real-time computer under a full operating system, and high-speed data retrieval at a remote location.
SUBJECT INDEX

A

Adaptive Filtering Technique for Digital Simulation 6226
—see also Control, Patterns

Adders

Error-Detecting Binary Adder 6146

Algebra

Ternary Switching Algebra for Combinational Switching Circuits 6131
—see also Matrices

Algorithms

Pattern-Classification Algorithms 6212

Algorithms for:

Prime-Implicant Generation Using Ternary Encoding and Decimal Arithmetic 6128

Graph-Theoretical Solution of Sequential Boolean Equations 6129

Finding Secondary Assignments of Synchronous Sequential Circuits 6139

Single Shift-Register Realizations for Sequential Machines 6140

Logical Multiplication on Computers with Two’s Complement Binary Arithmetic 6144

Residue Code Detection of Errors in Binary Adders 6146

Estimation of Jump Frequency for Program Segmentation 6181

Partial Ordering of the Syntax Elements of a Language Using Graph Theory 6208

Computation of Best Spline Approximations 6217

Computer Construction of PERT and CPM Project Networks 6233

Alphanumeric

—see Characters

Analog Computers

Computing Capabilities of Combinations of Digital and Analog Computers 6231
—see also Hybrid Computers

Approximation

Computation of Best Spline Approximations 6217
—see also Interpolation

Arithmetic

Prime-Implicant Generation Using Ternary Encoding and Decimal Arithmetic 6128

Logical Multiplication on Computers with Two’s Complement Binary Arithmetic 6144

Arithmetical Operations over Integers with Arbitrary Signs in Residue Class Number Systems 6145

Arithmetic Unit for Computing Discrete Fourier Transform of Audio Signals 6227
—see also Multiplication

Arrays

Application of Karnaugh Maps to Maitra Cascades in Cellular Logic Arrays 6132

Properties of Cellular Arrays for Logic and Storage 6153

Cellular Interconnection Arrays 6154

Large Scale Integration of MOS-Transistor Arrays and Laminated Ferrite Memory Plane 6159

Batch-Fabricated Magnetic-Film Arrays for the TX-2 Computer 6163

Balanced Capacitor READ-ONLY Storage Array 6164

Microcap Electron-Beam-Addressable Storage Elements and Arrays 6165

Assemblers

Assembly Program Under Control of 1108 EXEC 8 Data Handling Routines 6192

Associative

Magnetic Associative Memory 6157

Ordered Retrieval from Associative Memories 6184

Automata

Languages Accepted by Two-Way Balocon Automata 6134

Fuzzy Automata 6136

Languages, Automata and Classes of Chain-Encoded Patterns 6123
—see also Sequential

Automated; Automatic; Automation

Logic Design Automation System 6148

Automatic Error Detection and Self-Diagnosis 6152

Techniques for Automated Reading of Unformatted Text 6125

Batch Fabrication

Batch-Fabricated Magnetic-Film Arrays for the TX-2 Computer 6163

Batch Processing

Batch Processing Facility for Neurophysiological Investigations 6230

Boolean

Minimization of Boolean Function 6128

Graph-Theoretical Solution of Sequential Boolean Equations 6129

Karnaugh Map Solution of Boolean Equations 6130
—see also Switching Functions

C

Calculus

Interval Calculus for Matrices 6218
—see also Differential Equations, Integration

Cellular

Application of Karnaugh Maps to Maitra Cascades in Cellular Logic Arrays 6132

Properties of Cellular Arrays for Logic and Storage 6153

Cellular Interconnection Arrays 6154

Characters

MODEL T Display System for Characters and Graphics

Autonomous Character Recognition Machine 6214
—see also Patterns

Circuits

Trees and Matrices in Circuit Analysis Programs 6228
—see also Integrated Circuits, Logic, Networks, Sequential, Switching

Classification

Classification Scheme for Pattern Languages 6213
—see also Patterns

Codes; Encoding

Prime Implicant Generation Using Ternary Encoding and Decimal Arithmetic 6128

Residue Code Detection of Errors in Binary Adders 6146

Codes for Error Correction in Random-Access Memory Systems 6156

Languages, Automata and Classes of Chain-Encoded Patterns 6213

Combinational

Ternary Switching Algebra for Combinational Switching Circuits 6131

Communications

20A Data Communication Interface for the PDP-8 Computer 6176

Sorting Networks for Improving Communications in Computer Systems 6177
—see also Data Transmission, Man-Machine Communications

Compilers

Compiler for Efficient Execution of Concurrent Processes 6189

GIER ALGOL 4 Compiler System 6191

Use of Precompilers to Generate Special-Purpose FORTRAN Programs 6193

Computation; Computing

LISP A System for Incremental Computing 6194

Computing Capabilities of Combinations of Digital and Analog Computers 6231

Computational Linguistics

—see Languages

Computer Applications

System for Computer-Aided Selection and Assignment of Electronic Modules 6148

Recording Text in Computer-Usable Form with All Typographical Distinctions 6209

DAPLUS Language for Real-Time Information Retrieval from Hierarchical Data Bases 6211

Autonomous Character Recognition Machine 6214

Techniques for Automated Reading of Unformatted Text 6125

On-Line Random Number Generators 6225

Digital Modeling and Simulation of Nonlinear Systems 6226

Digital Processor for Generating Spectra in Real-Time 6227

Trees and Matrices in Circuit Analysis Programs 6228

Statistics Gathering and Simulation for the Apollo Real-Time Operating System 6229

On-Line Multiprocessing Interactive Computer System for Neurophysiological Investigations 6230

Computer Construction of PERT and CPM Project Networks 6233

Public Planning Information System and the Computer Utility 6234
—see also Control, Mathematics, Patterns Programs for, Retrieval

Computer Center Management

Methodology for Evaluating Time-Shared Computer System Usage 6203

Methodology for Calculating and Optimizing Real-Time System Performance and Cost 6232
Cost-Effectiveness Analysis in EDP System Selection 6235

Computer Industry
Market Dynamics of the Computer Industry 6126
Models of Order-Delivery Lags for Computers 6126

Computer Science
Master's Level Computer Science Curricula in U. S. Universities 6127

Computer Systems
Hierarchical Control Programs for Systems Evaluation 6151
Sorting Networks for Improving Communication in Computer Systems 6177
Multiple-Access Conversational Computer System 6205
Methodology for Calculating and Optimizing Real-Time System Performance and Cost 6232
Cost-Effectiveness Analysis in EDP System Selection 6235
—see also Digital Computers, Time-Sharing

Computers
Computer Capacity Trends 6126
Models of Order-Delivery Lags for Computers 6126
Organizations of a Fourth-Generation Computer 6141
Impact of Fourth-Generation Software on Hardware Design 6142
Programmed Word Length Computer 6143
Logical Multiplication on Computers with Two's Complement Binary Arithmetic 6144
Design of Self-Repairing Computers 6152
Batch Fabricated Magnetic Film Arrays for the TX-2 Computer 6163
Special-Purpose Time-Sharing Executive Program for the PDP-1 Computer 6174
201A Data Communication Interface for the PDP-8 Computer 6176
Interface for Connecting a PDP-8 Computer to a System/360 Model 67 6178
Programming Systems for Fourth-Generation Computers with Changeable Control Memories 6197
Multiple Operating System HYDRA for the EL X8 Computer 6200
—see also Analog Computers, Computer Systems, Digital Computers, Graphics, Time-Sharing

Control
Hierarchical Control Programs for System Evaluation 6151

Conversion
High-Speed Binary-to-Decimal Conversion 6147

Costs
Methodology for Calculating and Optimizing Real-Time System Performance and Cost 6232
Cost-Effectiveness Analysis in EDP System Selection 6235

Data
Sylvania Data Tablet for Graphic Data Input to Computers 6169
Graphical Data Management in a Time-Shared Environment 6170
Multilevel Graphical Data Structure 6173
201A Data Communication Interface for the PDP-8 Computer 6176
Description of a Set-Theoretic Data Structure 6185
Treatment of Hierarchical Data Structures in the SDC Time-Shared Management System 6186
Design of Programming Systems for Managing Large Data Files 6186
Automatic Association of Related Data Using Repeating Groups 6186
Computer Input to Data Bases 6188
Assembly Program Under Control of 1108 exec 8 Data Handling Routines 6192
DATAPLUS Language for Real-Time Information Retrieval from Hierarchical Data Bases 6211
—see also Files, Trees

Data Processing
—see Computer Systems, Digital Computers, Multiprocessing, Processing

Data Transmission
—see Codes, Communications, Interfaces

Decomposition
Krohn-Rhodes Machine Decomposition Theorem 6138

Design
Impact of Fourth-Generation Software on Hardware Design 6142
Design of Self-Repairing Computers 6152
—see also Logic Design

Diagnosis
Automatic Error Detection and Self-Diagnosis 6152
Diagnosis and Testing of Cellular Arrays 6153
—see also Error, Failures, Tests

Differential Equations (Ordinary)
Differential-Equation Method of Determining Intersection Points of Plane Curves 6220

Differential Equations (Partial)
Numerical Solution of Hyperbolic Partial Differential Equations by the Method of Characteristics 6222

Digital Computers
Comparison of Fault Simulation Methods for Digital Systems 6150
Computing Capabilities of Combinations of Digital and Analog Computers 6231
—see also Computer Systems, Processing Displays

Displays
DISPLAY System for Man-Machine Interaction 6170
MODEL T Display System for Characters and Graphics 6171
DX-1 Display System for Computer Entry and Manipulation of Hierarchical Graphical Information 6172, 6173, 6174, 6175
—see also Characters, Graphics

Electron Beams
Microcap Electron-Beam-Addressable Storage Elements and Arrays 6165

Equations
—see Algebra, Boolean, Differential Equations, Integral Equations

Error
Error-Detecting Binary Adder 6146

Automatic Error Detection and Self-Diagnosis 6152
Error Estimate of Runge-Kutta Solution of Systems of Differential Equations 6221
—see also Codes, Failures, Redundancy

Executive Systems
Special-Purpose Time-Sharing Executive Program for the PDP-1 Computer 6174
Monitor and Executive of the Berkeley Time-Sharing System 6204
—see also Operating Systems

Faults
Comparison of Fault-Simulation Methods for Digital Systems 6150
—see also Diagnosis, Reliability

Files
Design of Programming Systems for Managing Large Data Files 6186
—see also Sorting

Filtering
Adaptive Filtering Technique for Digital Simulation

Fourier
Arithmetic Unit for Computing Discrete Fourier Transforms of Audio Signals 6227

Geometry
Differential-Equation Method of Determining Intersection Points of Plane Curves 6220

Graph Theory; Graphs
Graph-Theoretical Solution of Sequential Boolean Equations 6129
Graphical Method of Solving Storage Element Flip-Flop Input Equations 6130
Transformation of Program Schemes to Standard Form Using Graph Theory 6179
Segmentation of Programs Using Graph Theory 6180
Partial Ordering of the Syntax Elements of a Language Using Graph Theory 6208
Graph Theoretical Treatment of Relational Properties 6210

Graphics
Sylvania Data Tablet for Graphic Data Input to Computers 6169
Graphical Data Management in a Time-Shared Environment 6170
MODEL T Display System for Characters and Graphics 6171
DX-1 Display System for Computer Entry and Manipulation of Hierarchical Graphical Information 6172, 6173, 6174, 6175
Multilevel Graphical Data Structure 6173
—see also Displays

Hybrid Computers
Computing Capabilities of Combinations of Digital to Analog Computers 6231

Information
Measures, Models and Measurements for Time-Shared Information Processing Utilities 6207
Man-Machine Information System for
ABSTRACTS OF CURRENT COMPUTER LITERATURE

Storage and Manipulation of Relational Sentences 6210
Public Planning Information System and the Computer Utility 6234
— see also Codes, Data, Retrieval
Input; Output
Graphical Method of Solving Storage Element Flip-Flop Input Equations 6130
Sylvania Data Tablet for Graphic Data Input to Computers 6169
Computer Input to Data Bases 6188
EXEC $ Input-Output Interface for FORTRAN Programs 6192
— see also Displays
Integral Equations
Starting Method for the Numerical Solution of Volterra Integral Equations 6223
Integrated Circuits
Modular Realizations of Universal Logic Circuits Using Integrated Circuits 6155
Large-Scale Integration of MOS-Transistor Arrays and Laminated Ferrite Memory Plates 6159
Commercial 2D Magnetic Memory with Integrated Electronics 6161
Integration; Integrals
Convergence Improvement for Some Improper Integrals 6219
Interconnection — see Man-Machine Communications Interconnections
Cellular Interconnection Arrays 6154
— see also Integrated Circuits, Interfaces
Interfaces
201A Data Communication Interface for the PDP-8 Computer 6176
Sorting Networks for Improving Communications in Computer Systems 6177
Interface for Connecting a PDP-8 Computer to a System/360 Model 67 6178 exec $ Input-Output Interface for FORTRAN Y Programs 6192
Interpolation
Numerical Determination of Points of Inflection by Successive Interpolation 6215
Interpreters; Interpretive
Interpretive System for Interactive Use with Time-Shared Computers via Teletype or CRT 6196
Interpretation of Instruction Sets by Microprograms 6197
Languages; Linguistics
Languages Accepted by Two-Way Balloon Automata 6134
Unsolvability of the Equality Problem for e-Finite Languages 6135
Man–Machine Communication by Natural Language 6166
Algebraic Language Design for Concurrent Processes 6189
Transformational Grammar Tester 6196 exec$—Command Language of the Project GENIE Time-Sharing Operating System 6205
Partial Ordering of the Syntax Elements of a Language Using Graph Theory 6208
DATAPLUS Language for Real-Time Information Retrieval from Hierarchical Data Bases 6211
Classification Scheme for Pattern Languages 6213
Application of Linguistic Theory to Pattern Analysis and Recognition 6213
— see also Programming Languages
Lists
FORMULA ALGOL Incorporating Formula Manipulation and List Processing 6190
LISP System for Incremental Computing 6194
Logic
Application of Karnaugh Maps to Maitra Cascades in Cellular Logic Arrays 6132
Logic Multiplication on Computers with Two’s Complement Binary Arithmetic 6144
Properties of Cellular Arrays for Logic and Storage 6153
Universal Logic Modules 6153
Programmed-Logic Arrays 6153
Modular Realizations of Universal Logic Circuits Using Integrated Circuits 6155
Graph Theoretical Treatment of Relational Properties 6210
— see also Switching Functions
Logic Design
Logic Design Automation System 6148
M
Magnetic Storage
Magnetic Cores
Fast Core Mass Memory with 2D Organization 6160
Commercial 2D Magnetic Memory with Integrated Electronics 6161
Heat of Ferrite Memory Cores During Operation 6162
Magnetic Films
Large Capacity Magnetic-Film Memory 6163
Majority
Approximate Reliability of Triply Redundant Majority-Voted Systems 6149
Man–Machine Communications (Interaction, Systems)
Man–Machine Communication by Natural Language 6166
Machine-to-Man Communication by Speech 6167, 6168
Graphic Man–Machine Communication 6169
DISPLAY System for Man–Machine Interaction 6170
Interpretive System for Interactive Use with Time-Shared Computers via Teletype or CRT 6196
Multiple-Access
Multiple-Access Conversational Computer System 6205
— see also Multiprocessing, Multiprogramming, Time-Sharing
Multiplication
Logical Multiplication on Computers with Two’s Complement Binary Arithmetic 6144
Multiprocessing
Avoiding Deadlock in Multitasking Operating Systems 6199
On-Line Multiprocessing Interactive Computer System for Neuropsychological Investigations 6230
— see also Multiprogramming, Time-Sharing
Multiprogramming
Testing the Performance of Systems in a Multiprogramming Environment 6151
Machine Organization for Multiprogramming 6201
Behavior of Programs in a Multiprogramming Paging Environment 6202
— see also Time-Sharing
N
Networks
Modular Tree Networks 6133
Sorting Networks for Improving Communications in Computer Systems 6177
Computer Construction of PERT and
On-Line Real-Time Paging Patterns

Noise
-see Noise

High-Speed Dataplus

On-Line Statistics Methodology

Hierarchical Control Multiple EXECUTIVE-Command Statistics
-see also Hierarchical Control Multiple EXECUTIVE-Command Statistics

Parallelism Manipulation

Behavior of Programs in a Multiprogramming Paging Environment 6201
-see also Multiprogramming Paging Environment 6202

Parallelism Control of Sequence and Parallelism in Modular Program 6182
Parallelism in Programs 6183

Patterns Manipulation of Tree Structures by Pattern Matching/Pattern Changing Rules 6196

Pattern-Classification Algorithms 6212
Languages, Automata and Classes of Chain-Encoded Patterns 6213
-see also Languages, Automata and Classes of Chain-Encoded Patterns 6213

Priority Turnaround Time for Messages of Differing Priorities in Teleprocessing Systems 6198

Processes, Processors Algebraic Language Design for Concurrent Processes 6189
Digital Processor for Generating Spectra in Real-Time 6227
-see also Algebraic Language Design for Concurrent Processes 6189
Digital Processor for Generating Spectra in Real-Time 6227

Programming
Programmed Word Length Computer 6143
Programmed-Logic Arrays 6153
-see also Parallelize-Logic Arrays 6153

Programming Languages
FORMULA ALGOL Incorporating Formula Manipulation and List Processing 6180
GIER ALGOL 4 Compiler System 6191
EXEC 8 Input-Output Interface for FORTRAN V Programs 6192
Use of Precompilers to Generate Special-Purpose FORTRAN Programs 6193
LISP System for Incremental Computing 6194
Formal Definition of PL/I and Other Programming Languages 6195
Programming-Language-Oriented Instruction Streams 6197

Programs Hierarchical Control Programs for Systems Evaluation 6151
Avoiding Deadlock in Multitasking Operating Systems 6199
Multiple Operating System HYDRA for the ELX 8 Computer 6200
EXECUTE—Command Language of the Project GENIE Time-Sharing Operating System 6205
Statistics Gathering and Simulation for the Apollo Real-Time Operating System 6229
-see also Executive Systems, Monitor Systems

Optical Design of Electron-Optical Systems for Scanning High-Density Storage Surfaces 6165

Optimization Methodology for Calculating and Optimizing Real-Time System Performance and Cost 6232
-see also Minimization

Paging Behavior of Programs in a Multiprogramming Paging Environmen 6202
-see also Multiprogramming Paging Environment 6202

Parallelism Control of Sequence and Parallelism in Modular Program 6182
Parallelism in Programs 6183

Patterns Manipulation of Tree Structures by Pattern Matching/Pattern Changing Rules 6196

On-Line Random Number Generators 6225
Computer Construction of PERT and CPM Project Networks 6233

Queues Turnaround Time for Messages of Differing Priorities in Teleprocessing Systems 6198

Random Processes
-see Noise, Numbers, Stochastic Processes

Real-Time
-see On-Line Recognition

Techniques for Automated Reading of Unformatted Text 6215
-see also Characters, Patterns

Redundancy Approximate Reliability of Triply Redundant Majority-Voted Systems 6149

Regular Generalized Transition Matrices, Regular Expressions and Regular Events 6137
Equivalence of Star-Free and Group-Free Regular Events 6138

Reliability Approximate Reliability of Triply Redundant Majority-Voted Systems 6149

Repairing Design of Self-Repairing Computers 6152

Retrieval Ordered Retrieval from Associative Memories 6184
Dataplus Language for Real-Time Information Retrieval from Hierarchical Data Bases 6211

Segmentation
Segmentation of Programs Using Graph Theory 6180
Estimation of Jump Frequency for Program Segmentation 6181
-see also Paging

Sequences
Statistical Properties of Random Digital Sequences 6224

Sequential
Graph-Theoretical Solution of Sequential Boolean Equations 6129
Unsolvability of the Equivalence Problem for Lambda-Free Nondeterministic Generalized Machines 6135
General Formulation of Sequential Machines 6136

Generalized Transition Matrices of Sequential Machines 6137
Krohn—Rhodes Machine Decomposition Theorem 6138
Finding Secondary Assignments of Synchronous Sequential Circuits 6139

Single Shift-Register Realizations for Sequential Machines 6140
Description of Noise Processes in Sequential Networks 6224
-see also Automata

Shift Registers
Single Shift-Register Realizations for Sequential Machines 6140
ABSTRACTS OF CURRENT COMPUTER LITERATURE

Simulation
Comparison of Fault-Simulation Methods for Digital Systems 6150
Digital Modeling and Simulation of Nonlinear Systems 6226
Statistics Gathering and Simulation for the Apollo Real-Time Operating System 6229

Software
Impact of Fourth-Generation Software on Hardware Design 6142
—see also Programming, Programs

Sorting
Sorting Networks for Improving Communications in Computer Systems 6177
Generalized Partial Block Sort 6187

Speech
Machine-to-Man Communication by Speech 6165, 6168
Synthesis of Speech 6167, 6168

Statistics; Statistical
Statistical Properties of Random Digital Sequences 6224
Statistics Gathering and Simulation for the Apollo Real-Time Operating System 6229
—see also Stochastic Processes

Stochastic Processes
Statistical Properties of Random Digital Sequences 6224
On-Line Random Number Generators 6225
—see also Noise

Storage; Memory
Graphical Method of Solving Storage Element Flip-Flop Equations 6130
Minimization of Wasted Storage 6143
Properties of Cellular Arrays for Logic and Storage 6153
Codes for Error Correction in Random-Access Memory Systems 6156
Review and Evaluation of Ferrite Memories 6158
MOS-Transistor-Driven Laminated Ferrite Memory 6159
Fast Core Mass Memory with 2D Organization 6160
Commercial 2D Magnetic Memory with Integrated Electronics 6161
Heating of Ferrite Memory Cores During Operation 6162
Large-Capacity Magnetic-Film Memory 6163
Balanced Capacitor Read-Only Storage Array 6164
High-Information-Density Storage Surfaces 6165
Microcap Electron-Beam-Addressable Storage Elements and Arrays 6165
Programming Systems for Fourth-Generation Computers with Changeable Control Memories 6197
Time-Sharing System with Memory Relabeling for Experimentation in Machine Interaction 6206
Man-Machine Information System for Storage and Manipulation of Relational Sentences 6210
—see also Arrays, Associative, Paging, Segmentation

Switching; Switches
Graphical Method of Solving Storage Element Flip-Flop Input Equations 6130
Ternary Switching Algebra for Combinational Switching Circuits 6131
—see also Automata, Sequential

Switching Functions
Prime-Implicant Generation Using Ternary Encoding and Decimal Arithmetic 6128
—see also Automata, Logic, Sequential

Synchronous
Finding Secondary Assignments of Synchronous Sequential Circuits 6139

Systems
Testing the Performance of Systems in a Multiprogramming Environment 6151
Digital Modeling and Simulation of Nonlinear Systems 6226
—see also Communications, Computer Systems, Multiprocessing, Multiprogramming, Operating Systems, Programming Systems, Time-Sharing

Teleprocessing
Turnaround Time for Messages of Different Priorities in Teleprocessing Systems 6198

Teletype
Interpretive System for Interactive Use with Time-Shared Computers via Teletype or CRT 6196

Ternary
Prime-Implicant Generation Using Ternary Encoding and Decimal Arithmetic 6128
Ternary Switching Algebra for Combinational Switching Circuits 6131

Tests; Evaluation
Hierarchical Control Programs for Systems Evaluation 6151
Testing Performance of Systems in a Multiprogramming Environment 6151
Methodology for Evaluating Time-Shared Computer System Usage 6203
Methodology for Calculating and Optimizing Real-Time System Performance and Cost 6232
—see also Diagnosis

Text
Recording Text in Computer-Usable Form with All Typographical Distinctions 6209
Techniques for Automated Reading of Unformatted Text 6215

Time-Sharing
Time-Sharing and Multiprogramming on a Fourth Generation Computer 6141
Graphical Data Management in a Time-Sharing Environment 6170
Special-Purpose Time-Sharing Executive Program for the PDP-1 Computer 6174
Treatment of Hierarchical Data Structures in the SDC Time-Shared Management System 6186
Interpretive System for Interactive Use with Time-Shared Computers via Teletype 6196
Methodology for Evaluating Time-Shared Computer System Usage 6203
Monitor and Executive of the Berkeley Time-Sharing System 6204
EXECUTIVE—Command Language of the Project GENIE Time-Sharing Operating System 6205
Time-Sharing System with Memory Relabeling for Experimentation in Machine Interaction 6206
Measures, Models and Measurements for Time-Shared Information Processing Utilities 6207
—see also Multiple-Access, Multiprogramming, Teleprocessing

Trees
Modular Tree Networks 6133
Manipulation of Tree Structures by Pattern Matching/Pattern Changing Rules 6196
Tries and Matrices in Circuit Analysis Programs 6228

Utility
Measures, Models and Measurements for Time-Shared Information Processing Utilities 6207
Public Planning Information System and the Computer Utility 6234
—see also Time-Sharing
AUTHOR INDEX

A
Abbas, S. A. 6164
Abraham, F. D. 6230
Abramson, H. D. 6208
Agnew, P. W. 6152
Agrawala, A. K. 6212
Allen, J. 6168
Anderson, A. H. 6163
Apostolatos, N. 6218
Ayling, J. K. 6164

B
Ballot, M. H. 6126
Bandak, K. 6195
Basu, S. K. 6179
Batcher, K. E. 6177
Bayes, A. 6187
Berztiss, A. T. 6180
Betyar, L. 6230
Bingham, H. W. 6183
Bleier, R. E. 6186
Bohi, M. J. 6142
Booth, T. L. 6224
Bowman, S. 6170
Bray, T. A. 6225
Brockhaus, M. 6200
Brown, K. A. 6232
Byrne, G. D. 6223

C
Casey, R. G. 6214
Chai, A. S. 6221
Chang, H. Y. 6150
Chase, E. N. 6172, 6173, 6174, 6175
Chelnokov, N. I. 6231
Childs, D. L. 6185
Coffman, E. G. 6202
Cone, D. R. 6165
Constantine, L. L. 6182
Crowther, T. S. 6163

D
Davies, W. A. 6140
Deutsch, L. P. 6204
Dietmeyer, D. L. 6128
Duley, J. R. 6128
Durham, L. 6204, 6205

E
Earley, J. 6190
Ehrman, J. R. 6144
Elliott, R. W. 6127
Elsap, B. 6153
Enea, H. J. 6189
Estrin, G. 6207
Etherton, M. 6205
Even, S. 6129

F
Fantauzzi, G. 6132
Feder, J. 6213
Feldman, A. 6188
Feng, T. M. 6157
Ferguson, R. 6222
Fisher, A. C. 6233
Forbes, R. E. 6152

G
Gaddess, T. G. 6146
Gifford, C. E. 6164
Ginsburg, S. 6134
Glau, R. G. 6164
Gold, M. M. 6203
Gordon, I. 6159
Gostelow, K. P. 6148
Griffiths, T. V. 6135

H
Haratz, D. 6130
Harvey, R. L. 6159
Hauth, C. 6198
HAVender, J. W. 6199
Herndon, T. O. 6163
Herrmann, R. L. 6131
Hertel, H. F. 6229
Heynick, L. N. 6165
Ho, Y.-C. 6212
Hopcroft, J. 6134

J
Jarratt, P. 6216
Johnston, R. 6230
Jones, R. D. 6234
Jordan, Jr., W. F. 6161

K
Kameda, T. 6137
Kamman, A. B. 6236
Kautz, W. H. 6153, 6154
Kay, M. 6209
Keele, D. D. 6151
Kleinrock, L. 6207
Knight, K. E. 6126
Král, J. 6181
Kuiken, H. K. 6220
Kulisch, U. 6218
Kwei, T. C. 6164

L
Lampson, B. W. 6204
Lass, E. 6141
Lawson, Jr., H. W. 6197
Lee, F. F. 6167
Lee, J. 6222
Levitt, K. N. 6154
Licktenberger, W. W. 6206
Lickhalter, R. A. 6170
Liebman, J. S. 6233
Lucky, A. L. 6143

M
Maywald, W. 6162
Mann, W. C. 6210
Manning, E. G. 6150
Marsaglia, G. 6225
McWilliams, G. V. 6219
Meo, A. R. 6133
Meyer, A. R. 6129, 6138
Miller, J. C. 6171
Mills, D. 6178
Moss, H. I. 6159

N
Nagy, G. 6214, 6215
Naur, P. 6191
Nemhauser, G. L. 6233

O
O'Neill, J. C. 6223
Orlow, T. 6222

P
Pierce, J. R. 6166
Pirtle, M. W. 6206
Porter, J. D. 6235

R
Raffel, J. I. 6163
Reichard, R. W. 6161
Reigel, E. W. 6183
Robbi, A. D. 6159
Roberts, K. T. 6165
Rokso, J. S. 6226
Rubin, D. K. 6249
Rudick, B. H. 6235

S
Sallen, R. P. 6169
Sandewall, E. J. 6194
Santos, E. S. 6136
Saxton, D. R. 6236
Schild, A. 6210
Schmocker, M. S. 6147
Schoene, W. J. 6196
Schumaker, L. L. 6217
Schuur, C. M. 6160
Sedore, S. R. 6228
Shahbender, R. 6158
Shively, R. R. 6227
Sinowitz, N. R. 6211
Sirmay, I. A. 6231
Sobornykov, Yu. P. 6145
Srinivasan, C. V. 6156
Stanley, W. 1. 6229
Stieglitz, C. B. 6152
Stimler, S. 6222
Stone, H. S. 6153

T
Tang, C. K. 6155
Taren, W. J. 6164
Teixeira, J. F. 6169
Tesler, L. G. 6189
Thompson, R. W. 6219
Tomg, H. C. 6139
Tsivetaeva, I. L. 6231
Tuska, J. W. 6159

V
Varian, L. C. 6202

W
Waksman, A. 6154
Wallmark, J. T. 6159
Walter, A. B. 6142
Walter, C. J. 6142
Wee, W. G. 6136
Wegner, P. 6201
Wentworth, C. 6159
Werts, H. J. 6193
Westlake, D. A. 6172, 6173, 6174, 6175
Wettstein, H. 6200
Wine, C. M. 6171
Wolinsky, A. 6184
Wood, D. E. 6176
Woodward, C. E. 6163

Y
Yau, S. S. 6155

Z
Ziehe, T. 6209