$$v_1 = m_{11}u_1 + m_{12}u_2 + \cdots + m_{1n}u_n$$
$$v_2 = m_{21}u_1 + m_{22}u_2 + \cdots + m_{2n}u_n$$
$$\vdots$$
$$v_n = m_{n1}u_1 + m_{n2}u_2 + \cdots + m_{nn}u_n$$

or

$$[v_i] = [m_{ij}][u_j].$$

It is then stated without qualification that $[m_{ij}]$ is nonsingular. It might have been mentioned that this followed from the linear independence of the elements of a set of basis vectors. It is inferred that, having chosen two sets of basis vectors, the elements of the matrix M may be obtained from the solution of the above set of linear equations.

It is then shown that if the coordinates of a vector v are c_i with respect to a basis set v_i, $i=1, 2, \cdots, n$, the vector may be written as

$$v = [c_i][v_i].$$

It is then said to be seen that the coordinates (r_j) or v with respect to the basis vectors $\{u_i\}$, $i=1, 2, \cdots, n$ are given by the equation

$$[r_j] = [c_i][m_{ij}].$$

This might have been more immediately apparent if the equation

$$v = [r_j][u_j]$$

had been included. The development would then have been

$$v = [c_i][v_i]$$
$$v_i = [m_{ij}][u_j]$$
$$v = [c_i][m_{ij}][u_j]$$
$$v = [r_j][u_j]$$
$$\therefore [r_j] = [c_i][m_{ij}].$$

Linear sequential machines are treated in Chapter VIII following mathematical formulation in Chapters II and VII. There are good arguments for connecting this material with the chapter on regular expressions, and for this reason additional sections concerning the synthesis of linear (or polylinear) machines from regular expressions could be included. The material concerning the relationships between the rank of the K_k matrix and minimal state machines is quite simple for those readers having a basic knowledge of matrix theory; however, the rules applied are not explained, at least not explicitly, in the referenced chapter.

The first portion of Chapter IX gives a "mechanical" description of a Turing machine, showing the basic operation, and then goes into a formal mathematical definition. The next two sections present algorithmic approaches to accomplish large computational problems. Several very elementary submachines are first given and a method is given to accomplish the necessary interconnect for larger submachines that may be used to perform basic information-processing operations. Then, machines are formed to perform such operations as addition, subtraction, and multiplication. It is also shown that composition, primitive recursion, and minimalization, which can be performed by such machines, can be used to generate a large variety of more complex functions. The next two sections are devoted to a detailed treatment of compatibility. The last portion of the chapter is devoted to special Turing machines, with multiple tapes for separate storage and processing of information, or with special rules governing how information can be taken from and put into the external memory.

In Chapter X, regular expression is again a topic, but is now viewed from a different (and broader) perspective. The discussion of artificial languages in just one chapter must, of course, be limited. Chapter X deals with phrase-structure languages because they are closely related to the systems covered in preceding chapters. The chapter is divided into four sections: 1) definitions and terminology, 2) language-machine relationships, 3) operations on languages, and 4) decision problems. The first two sections establish what an artificial language is and how a language is associated with a particular machine. The last two sections extend further the mathematical basis of artificial languages.

Chapters XI and XII are devoted to random sequences and random processes in sequential machines.

A substantial list of references is included at the end of each chapter and challenging problems (with solutions in Appendix IV) are given at the end of each section. There are also problems given at the end of most of the chapters to point out the salient fundamentals included in the chapters.

The material in the book is organized so that it can be used for a single course, or for a two-course sequence without loss of continuity. It is, however, unfortunate that so many printing errors were made in the first printing.

The author has, in the opinion of this reviewer, accomplished a very commendable contribution.

CHESTER C. CARROLL
and his students
Dept. of Elec. Engrg.
Auburn University
Auburn, Ala. 36830

B. EXECUTIVE PROGRAM DEVELOPMENT

R68-41 Development of Executive Routines, Both Hardware and Software—Albert B. Tonik (*1967 Fall Joint Computer Conf., AFIPS Proc.*, vol. 31. Washington, D. C.: Thompson, 1967, pp. 395–408).

Innumerable papers by Russian authors using probabilistic models for physical systems are prefaced by a comment explaining that the results obtained must follow from dialectic materialism, and therefore the same results could in principle be derived deterministically as well as statistically, save for mathematical convenience. A similarly political motivation may unconsciously underlie many papers in the computer field which explain that ever more elaborate executive programs are motivated exclusively by a desire to increase hardware efficiency and improve system economy.

Thus this paper, an otherwise excellent review of the array of problems encountered in organizing an executive program, weakens its semi-historical development by presenting each elaboration of executive responsibility as motivated only by improved hardware utilization. In the most serious example of the distortion caused by this viewpoint, the paper suggests that the motivation for paging as a scheme for allocating primary memory is to sandwich additional programs into memory in the holes left between other programs. Whether or not paging increases or decreases the number of words of primary memory which can be utilized probably depends on the statistics of the programs involved. The more serious issues (and the ones which raise the most emotional arguments) are the abilities of a paged machine to present the programmer with an interface which does not change every time a new box of memory is added, and to relieve the programmer of explicit overlay organization. The author mentions these latter issues as though they were virtually side effects of the effort to increase hardware utilization.

Aside from this single significant complaint, this paper is a welcome relief from the long series of "here is how we implemented a multi——— system on the ——— computer." The author has taken the trouble to identify many of the sources of difficulty and complexity in an executive program. This is one of the few papers one can point to which recognize the importance of error handling and program snapshot/restart procedures in their contribution to the complexity of executive programs.

A discussion of solutions as comprehensive as the author's discussion of problems would require at least a book. The author has taken what appeared to be an easy shortening scheme—he indicates one or two techniques of attacking each problem. Unfortunately, he becomes unintentionally dogmatic when he presents the technique he is most familiar with and fails to indicate that there may exist several other techniques to solve the same problem. This comment applies, for example, to his discussions of priority schemes, error handling, and interrupt handling.

Overall, the paper provides a lot of food for thought, and is well worth exploring before implementing another operation system. A comprehensive bibliography of papers on operating systems provides valuable reference.

J. H. Saltzer
Waban, Mass.

The text of this article shows a wide variation in communication quality: portions of it are not to be approached casually.

Beverly F. Char
Codie S. Wells
The MITRE Corporation
Bedford, Mass.

C. FILE RECORD DESIGN

R68-42 On Designing Generalized File Records for Management Information Systems—Frank H. Benner (*1967 Fall Joint Computer Conf., AFIPS Proc.*, vol. 31. Washington, D. C.: Thompson, 1967, pp. 291–303).

The author has chosen a somewhat inappropriate title for this paper; the file records he discusses are not "generalized," but rather *tailored* to predictable real world constraints, and the "management information systems" he considers form only a portion of the spectrum. Nevertheless, he presents an objective technique for file record design quite applicable in certain cases. Given 1) known and predictable data and basic interrelationships subject to 2) known and predictable use (processing/querying), then the design techniques presented by the author should certainly ease the designer's conscience, if nothing else, and will probably materially assist in "tuning" a record-oriented data management system. The analytic design techniques presented form one element in a suggested iterative design loop "of data collection, preliminary file design, simulate (sic), analyze (sic) results, ..." Additional consideraton of "tuning" a system might well lead to consideration of special instrumentation, special reduction for the data gathered by such instrumentation, adaptive systems which "tune" themselves for optimum performance, and other subjects. This article concentrates on the analytic design techniques, to the exclusion of the other aspects of the complex design cycle.

The author discusses techniques which require a rather large investment in manpower to apply, even if one assumes that "most designers will have a linear programming package available to them, or can have one written," as does Benner. The techniques presented have application to data/procedure sets in which large quantities of relatively predictable data are processed in a relatively predictable fashion. Since the techniques are strictly record-oriented, they are not directly applicable to some data management systems (e.g., TDMS,[1] ADAM[2]), which are object-oriented, or which use list structures. The "management" being serviced by systems amenable to Benner's techniques is more probably lower management, rather than middle or upper management—the information requirements of the latter groups tend to be less predictable than those of the former.

[1] A. H. Vorhaus and R. D. Wills, "A time shared data management system: a new approach to data management," Systems Development Corporation, Binghamton, N. Y., SP-2747, February 13, 1967.
[2] T. L. Connors, "ADAM: a generalized data management system," the MITRE Corporation, Bedford, Mass., MTP-29, April 1966; also *1966 Spring Joint Computer Conf., AFIPS Proc.*, vol. 28. Washington, D. C.: Thompson, 1966, pp. 193–203.

D. CONTINUOUS SYSTEM MODELING PROGRAMS

R68-43 Two Continuous System Modelling Programs—R. D. Brennan and M. Y. Silberberg (*IBM Sys. J.*, vol. 6, no. 4, pp. 242–266, 1967).

This paper concerning two IBM programs was written by two IBMers for an IBM publication. This might given reason to question the objectivity of the paper and its value to the general community. In actuality, though, the two programs discussed are of sufficient general interest, if for no other reason than the popularity of the machines for which they have been implemented, and the authors have maintained sufficient perspective to warrant reading of the paper by all persons interested in continuous system stimulation.

The paper presents the two IBM continuous system modeling programs:

1) the 1130 CSMP—a block-diagram-oriented simulation program which is a direct descendant of the PACTOLUS program for the IBM 1620, and

2) System/360 CSMP—an equation-oriented simulation language which appears to be a descendant of DSL/90 for the IBM 7090 and is closely related to CSSL,[1] the SCI recommended standard continuous system simulation language.

In general, the treatment of historical background is excellent, but it tends to slight the importance of some of the most recent developments. In particular, the relation of System/360 CSMP to CSSL is not mentioned even though one of the authors is a member of the SCI Simulation Software Committee and had access to all the documents and discussion in the development of the CSSL report. The sections describing the CSMP programs are well written and the examples provide the reader with a good understanding of the use of the programs. Of particular interest is the application of both a block-oriented (1130 CSMP) and an equation-oriented (360 CSMP) simulation language to the same problem. This example and the associated discussion help to illustrate the relative strengths and weaknesses of the two approaches.

In summary, although the paper has aspects of an "in-house" report, the material and its presentation are worthy of a wider distribution.

J. C. Strauss
Dept. of Elec. Engrg.
Carnegie-Mellon University
Pittsburgh, Pa.

[1] SCI Simulation Software Committee, J. C. Strauss, Ed., "The SCI continuous system simulation language (CSSL)," *Simulation*, vol. 9, no. 6 pp. 281–303, 1967.