CURRICULUM 68

Recommendations for Academic Programs in Computer Science

A REPORT OF THE ACM CURRICULUM COMMITTEE ON COMPUTER SCIENCE

Dedicated to the Memory of Silvio O. Navarro

This report contains recommendations on academic programs in computer science which were developed by the ACM Curriculum Committee on Computer Science. A classification of the subject areas contained in computer science is presented and twenty-two courses in these areas are described. Prerequisites, catolog descriptions, detailed outlines, and annotated bibliographies for these courses are included. Specific recommendations which have evolved from the Committee's 1965 Preliminary Recommendations are given for undergraduate programs. Graduate programs in computer science are discussed, and some recommendations are presented for the development of master's degree programs. Ways of developing guidelines for doctoral programs are discussed, but no specific recommendations are made. The importance of service courses, minors, and continuing education in computer science is emphasized. Attention is given to the organization, staff requirements, computer resources, and other facilities needed to implement computer science educational programs.

KEY WORDS AND PHRASES: computer science courses, computer science curriculum, computer science education, computer science academic programs, computer science graduate programs, computer science undergraduate programs, computer science course bibliographies

CR CATEGORIES: 1.52

Preface

The Curriculum Committee on Computer Science (C^3S) was initially formed in 1962 as a subcommittee of the Education Committee of the Association for Computing Machinery. In the first few years of its existence this subcommittee functioned rather informally by sponsoring a number of panel discussions and other sessions at various national computer meetings. The Curriculum Committee became an independent committee of the ACM in 1964 and began an active effort to formulate detailed recommendations for curricula in computer science. Its first report, "An Undergraduate Program in Computer Science—Preliminary Recommendations" [1], was published in the September 1965 issue of *Communications of the ACM*.

The work of the Committee during the last two years has been devoted to revising these recommendations on undergraduate programs and developing recommendations for graduate programs as contained in this report. The primary support for this work has been from the National Science Foundation under Grant Number GY-305, received in July 1965.

The Committee membership during the preparation of this report was:

William F. Atchison, University of Maryland (Chairman)
Samuel D. Conte, Purdue University
John W. Hamblen, SREB and Georgia Institute of Technology
Thomas E. Hull, University of Toronto
Thomas A. Keenan, EDUCOM and the University of Rochester
William B. Kehl, University of California at Los Angeles
Edward J. McCluskey, Stanford University
Silvio O. Navarro,* University of Kentucky
Werner C. Rheinboldt, University of Maryland
Earl J. Schweppe, University of Maryland (Secretary)
William Viavant, University of Utah
David M. Young, Jr., University of Texas

* Dr. Navarro was killed in an airplane crash on April 3, 1967.

In addition to these members many others have made valuable contributions to the work of the Committee. Their names and affiliations are listed at the end of this report. Robert Ashenhurst and Peter Wegner have given especial assistance in the preparation of this report.

CONTENTS

1. Introduction

Following the appearance of its Preliminary Recommendations [1], the Curriculum Committee on Computer Science received many valuable comments, criticisms, and suggestions on computer science education. From these, the advice of numerous consultants, and the ideas of many other people, the Committee has prepared this report, "Curriculum 68," which is a substantial refinement and extension of the earlier recommendations. The Committee hopes that these new recommendations will stimulate further discussion in this area and evoke additional contributions to its future work from those in the computing profession. The Committee believes strongly that a continuing dialogue on the process and goals of education in computer science will be vital in the years to come.

In its Preliminary Recommendations the Committee devoted considerable attention to the justification and description of "computer science" as a discipline. Although debate on the existence of such a discipline still continues, there seems to be more discussion today on what this discipline should be called and what it should include. In a recent letter [2], Newell, Perlis, and Simon defend the name "computer science." Others, wishing perhaps to take in a broader scope and to emphasize the information being processed, advocate calling this discipline "information science" [3] or, as a compromise, "the computer and information sciences" [4]. The Committee has decided to use the term "computer science" throughout this report, although it fully realizes that other names may be used for essentially the same discipline.

In attempting to define the scope of this discipline, the Committee split computer science into three major subject divisions to which two groups of related areas were then added. Using this as a framework, the Committee developed a classification of the subject areas of computer science and some of its related fields, and this classification is presented in Section 2.

As was the case for its Preliminary Recommendations, the Committee has devoted considerable effort to the development of descriptions, detailed outlines, and bibliographies for courses in computer science. Of the sixteen courses proposed in the earlier recommendations, eleven have survived in spirit if not in detail. Two of the other five courses have been split into two courses each, and the remaining three have been omitted since they belong more properly to other disciplines closely related to computer science. In addition seven new courses have been proposed, of which Course B3 on "discrete structures" and Course I3 on "computer organization" are particularly notable. Thus this report contains detailed information—in the form of catalog descriptions and prerequisites in Section 3 and detailed outlines and annotated bibliographies in the Appendix—on a total of twenty-two courses.

Another important issue which concerned the Curriculum Committee is the extent to which undergraduate programs as opposed to graduate programs in computer science ought to be advocated. Certainly, both undergraduate and graduate programs in "computer science" do now exist, and more such programs operate under other names such as "information science" or "data processing" or as options in such fields as mathematics or electrical engineering. A recent survey [5], supported by the National Science Foundation and carried out by the Computer Sciences Project of the Southern Regional Education Board, contains estimates of the number of such degree programs operating in 1964–1965 and projections of the number planned to be operating by 1968–1969. These estimates and projections can be summarized as follows:

Program name	*Program level*					
	Bachelor's		*Master's*		*Doctoral*	
	1964 1965	1968 1969	1964 1965	1968 1969	1964 1965	1968 1969
Computer Science	11	92	17	76	12	38
Data Processing	6	15	3	4	1	2
Information Science	2	4	12	17	4	13
Similar Programs	25	40	29	40	21	28

The information contained in these figures is interesting for two reasons. First, it shows that the number of computer science degree programs will continue to grow rapidly even if some of the programs now being planned do not come into being. Second, it shows a strong tendency to use the name "computer science," although the availability of academic work in computing is not limited to institutions having a department or a program operating under that title.

A major purpose of the Committee's recommendations on undergraduate programs and master's degree programs given in Sections 4 and 5 is to provide a sense of direction and a realizable set of goals for those colleges and universities which plan to provide computer science education for undergraduate and /or graduate students. The discussion in Section 6 of how guidelines for doctoral programs may be developed is very general, mainly because this is a difficult area in which to make detailed recommendations.

The importance of service courses, minors, and continued education in computer science has also been of concern to the Committee. Although detailed work still needs to be done, some preliminary discussion of the needs in these areas is given in Section 7. In Section 8 some of the problems of implementing an educational program in computer science are discussed.

In general the difficulties in establishing such programs are formidable; the practical problems of finding qualified faculty, of providing adequate laboratory facilities, and of beginning a program in a new area where there are few textbooks are severe. These problems are magnified for baccalaureate programs in comparison with graduate programs, where the admission can be more closely controlled.

The demand for substantially increased numbers of persons to work in all areas of computing has been noted in a report of the National Academy of Sciences-National Research Council [6] (commonly known as the "Rosser Report") and in a report of the President's Science Advisory Committee [7] (often called the "Pierce Report"). Although programs based on the recommendations of the Curriculum Committee can contribute substantially to satisfying this demand, such programs will not cover the full breadth of the need for personnel. For example, these recommendations are not directed to the training of computer operators, coders, and other service personnel. Training for such positions, as well as for many programming positions, can probably be supplied best by applied technology programs, vocational institutes, or junior colleges. It is also likely that the majority of applications programmers in such areas as business data processing, scientific research, and engineering analysis will continue to be specialists educated in the related subject matter areas, although such students can undoubtedly profit by taking a number of computer science courses.

In addition to this Committee, several other organizations have set forth guidelines to aid educational institutions in the establishment of programs pertinent to the needs of today's computer-oriented technology. Prominent among these are the reports of the Committee on the Undergraduate Program in Mathematics (CUPM) of the Mathematical Association of America [8], the COSINE Committee of the Commission on Engineering Education [9], and the Education Committee of the British Computer Society [10]. Also, the ACM Curriculum Committee on Computer Education for Management, chaired by Daniel Teichroew, is now beginning to consider educational matters related to the application of computers to "management information systems." The Curriculum Committee has benefited greatly from interchanging ideas with these other groups. In addition, the entire Committee was privileged to take part in "The Graduate Academic Conference in Computing Science" [11] held at Stony Brook in June 1967.

Computer science programs, in common with those of all disciplines, must attempt to provide a basis of knowledge and a mode of thinking which permit continuing growth on the part of their graduates. Thus, in addition to exposing the student to a depth of knowledge in computer science sufficient to lay the basis for professional competence, such programs must also provide the student with the intellectual maturity which will allow him to stay abreast of his own discipline and to interact with other disciplines.

2. Subject Classification

The scope of academic programs and curricula in computer science will necessarily vary from institution to institution as dictated by local needs, resources, and objectives. To provide a basis for discussion, however, it seems desirable to have a reasonably comprehensive system for classifying the subject areas within computer science and related fields. Although any such system is somewhat arbitrary, it is hoped that any substantial aspect of the computer field, unless specifically excluded for stated reasons, may be found within the system presented here. The subject areas within computer science will be classified first; those shared with or wholly within related fields will be discussed later in this section.

Computer Science. The subject areas of computer science are grouped into three major divisions: "information structures and processes," "information processing systems," and "methodologies." The subject areas contained in each of these divisions are given below together with lists of the topics within each subject area.

I. INFORMATION STRUCTURES AND PROCESSES

This subject division is concerned with representations and transformations of information structures and with theoretical models for such representations and transformations.

1. Data Structures: includes the description, representation, and manipulation of numbers, arrays, lists, trees, files, etc.; storage organization, allocation, and access; enumeration, searching and sorting; generation, modification, transformation, and deletion techniques; the static and dynamic properties of structures; algorithms for the manipulation of sets, graphs, and other combinatoric structures.

2. Programming Languages: includes the representation of algorithms; the syntactic and semantic specification of languages; the analysis of expressions, statements, declarations, control structures, and other features of programming languages; dynamic structures which arise during execution; the design, development and evaluation of languages; program efficiency and the simplification of programs; sequential transformations of program structures; special purpose languages; the relation between programming languages, formal languages, and linguistics.

3. Models of Computation: includes the behavioral and structural analysis of switching circuits and sequential machines; the properties and classification of automata; algebraic automata theory and model theory; formal languages and formal grammars; the classification of languages by recognition devices; syntactic analysis; formal

specification of semantics; syntax directed processing; decidability problems for grammars; the treatment of programming languages as automata; other formal theories of programming languages and computation.

II. INFORMATION PROCESSING SYSTEMS

This subject division is concerned with systems having the ability to transform information. Such systems usually involve the interaction of hardware and software.

1. Computer Design and Organization: includes types of computer structure—von Neumann computers, array computers, and look-ahead computers; hierarchies of memory—flip-flop registers, cores, disks, drums, tapes—and their accessing techniques; microprogramming and implementation of control functions; arithmetic circuitry; instruction codes; input-output techniques; multiprocessing and multiprogramming structures.

2. Translators and Interpreters: includes the theory and techniques involved in building assemblers, compilers, interpreters, loaders, and editing or conversion routines (media, format, etc.).

3. Computer and Operating Systems: includes program monitoring and data management; accounting and utility routines; data and program libraries; modular organization of systems programs; interfaces and communication between modules; requirements of multiaccess, multiprogram and multiprocess environments; large scale systems description and documentation; diagnostic and debugging techniques; measurement of performance.

4. Special Purpose Systems: includes analog and hybrid computers; special terminals for data transmission and display; peripheral and interface units for particular applications; special software to support these.

III. METHODOLOGIES

Methodologies are derived from broad areas of applications of computing which have common structures, processes, and techniques.

1. Numerical Mathematics: includes numerical algorithms and their theoretical and computational properties; computational error analysis (for rounding and truncation errors); automatic error estimates and convergence properties.

2. Data Processing and File Management: includes techniques applicable to library, biomedical, and management information systems; file processing languages.

3. Symbol Manipulation: includes formula operations such as simplification and formal differentiation; symbol manipulation languages.

4. Text Processing: includes text editing, correcting, and justification; the design of concordances; applied linguistic analysis; text processing languages.

5. Computer Graphics: includes digitizing and digital storage; display equipment and generation; picture compression and image enhancement; picture geometry and topology; perspective and rotation; picture analysis; graphics languages.

6. Simulation: includes natural and operational models; discrete simulation models; continuous change models; simulation languages.

7. Information Retrieval: includes indexing and classification; statistical techniques; automatic classification; matching and search strategies; secondary outputs such as abstracts and indexes; selective dissemination systems; automatic question answering systems.

8. Artificial Intelligence: includes heuristics; brain models; pattern recognition; theorem proving; problem solving; game playing; adaptive and cognitive systems; man-machine systems.

9. Process Control: includes machine tool control; experiment control; command and control systems.

10. Instructional Systems: includes computer aided instruction.

Related Areas. In addition to the areas of computer science listed under the three divisions above, there are many related areas of mathematics, statistics, electrical engineering, philosophy, linguistics, and industrial engineering or management which are essential to balanced computer science programs. Suitable courses in these areas should be developed cooperatively with the appropriate departments, although it may occasionally be desirable to develop some of these courses within the computer science program.

Since it is not feasible in this report to list all of the areas which might be related to a computer science program, let alone indicate where courses in these areas should be taught, the following listing is somewhat restricted. It is grouped into two major divisions: "mathematical sciences" and "physical and engineering sciences."

IV. MATHEMATICAL SCIENCES

1. Elementary Analysis
2. Linear Algebra
3. Differential Equations
4. Algebraic Structures
5. Theoretical Numerical Analysis
6. Methods of Applied Mathematics
7. Optimization Theory
8. Combinatorial Mathematics
9. Mathematical Logic
10. Number Theory
11. Probability and Statistics
12. Operations Analysis

V. PHYSICAL AND ENGINEERING SCIENCES

1. General Physics
2. Basic Electronics
3. Circuit Analysis and Design
4. Thermodynamics and Statistical Mechanics
5. Field Theory
6. Digital and Pulse Circuits
7. Coding and Information Theory
8. Communication and Control Theory
9. Quantum Mechanics

No attempt has been made to include within this classification system all the subject areas which make use of computer techniques, such as chemistry and economics; indeed, to list these would require inclusion of a major portion of the typical university catalog. Furthermore, the sociological, economic, and educational implications of developments in computer science are not discussed in this report. These issues are undoubtedly important, but they are not the exclusive nor even the major responsibility of computer science. Indeed, other departments such as philosophy and sociology should be urged to cooperate with computer scientists in the development of courses or seminars covering these topics, and computer science students should be encouraged to take these courses.

3. Description of Courses

The computer science courses specified in this report are divided into three categories: "basic," "intermediate," and "advanced." The basic courses are intended to be taught primarily at the freshman-sophomore level, whereas both the intermediate and the advanced courses may be taught at the junior-senior or the graduate level. In general, the intermediate courses are strongly recommended as part of undergraduate programs. The advanced courses are classified as such either because of their higher level of prerequisites and required maturity or because of their concern with special applications of computer science.

In addition to more elementary computer science courses, certain courses in mathematics are necessary, or at least highly desirable, as prerequisites for some of the proposed courses. More advanced mathematics courses may be included as supporting work in the programs of some students. Because of the considerable variation in the level and content of mathematics courses among (and even within) schools, the courses described by the Committee on the Undergraduate Program in Mathematics (CUPM) in the report, "A General Curriculum in Mathematics for Colleges" [12] have been used to specify the prerequisites for the proposed courses in computer science and requirements for degrees. Other pertinent mathematics courses are described in the CUPM reports, "Recommendations on the Undergraduate Mathematics Program for Engineers and Physicists" [13] and "A Curriculum in Applied Mathematics" [14].

The titles and numbers of all the courses proposed in this report and the pertinent courses recommended by CUPM are shown in Figure 1 along with the prerequisite structure linking these courses. The courses described below, which make up the core of the undergraduate program, are also singled out in Figure 1. The relatively strong prerequisite structure proposed for these core courses allows their content to be greatly expanded from what a weaker structure would permit. The Committee recognizes that other—perhaps weaker—prerequisite structures might also be effective and that the structure shown will change along with the course content as computer science education develops. Prerequisites proposed for the advanced courses are subject to modification based on many orientations which these courses may be given at individual institutions.

Most of the courses have been designed on the basis of three semester hours of credit. Laboratory sessions, in which the more practical aspects of the material can be presented more effectively than in formal lectures, have been included where appropriate. The proposed number of hours of lecture and laboratory each week and the number of semester hours of credit for the course are shown in parentheses in the catalog descriptions below. For example, (2-2-3) indicates two hours of lecture and two hours of laboratory per week for a total of three semester hours of credit.

Course Catalog Descriptions and Prerequisites

For each of the courses listed below, a brief statement on the approach which might be taken in teaching it is given in the Appendix along with the detailed outlines of its proposed contents and annotated bibliographies of pertinent source materials and textbooks.

☐The first course is designed to provide the student with the basic knowledge and experience necessary to use computers effectively in the solution of problems. It can be a service course for students in a number of other fields as well as an introductory course for majors in computer science. Although no prerequisites are listed, it is assumed that the student will have had a minimum of three years of high school mathematics. All of the computer science courses which follow will depend upon this introduction.

Course B1. Introduction to Computing (2-2-3)

Algorithms, programs, and computers. Basic programming and program structure. Programming and computing systems. Debugging and verification of programs. Data representation. Organization and characteristics of computers. Survey of computers, languages, systems, and applications. Computer solution of several numerical and nonnumerical problems using one or more programming languages.

☐The second course is intended to lay a foundation for more advanced study in computer science. By familiarizing the student with the basic structure and language of machines, the content of this course will give him a better understanding of the internal behavior of computers, some facility in the use of assembly languages, and an ability to use computers more effectively—even with procedure-oriented languages.

Course B2. Computers and Programming (2-2-3)

Prerequisite: Course B1.

Computer structure, machine language, instruction execution, addressing techniques, and digital representation of data. Computer

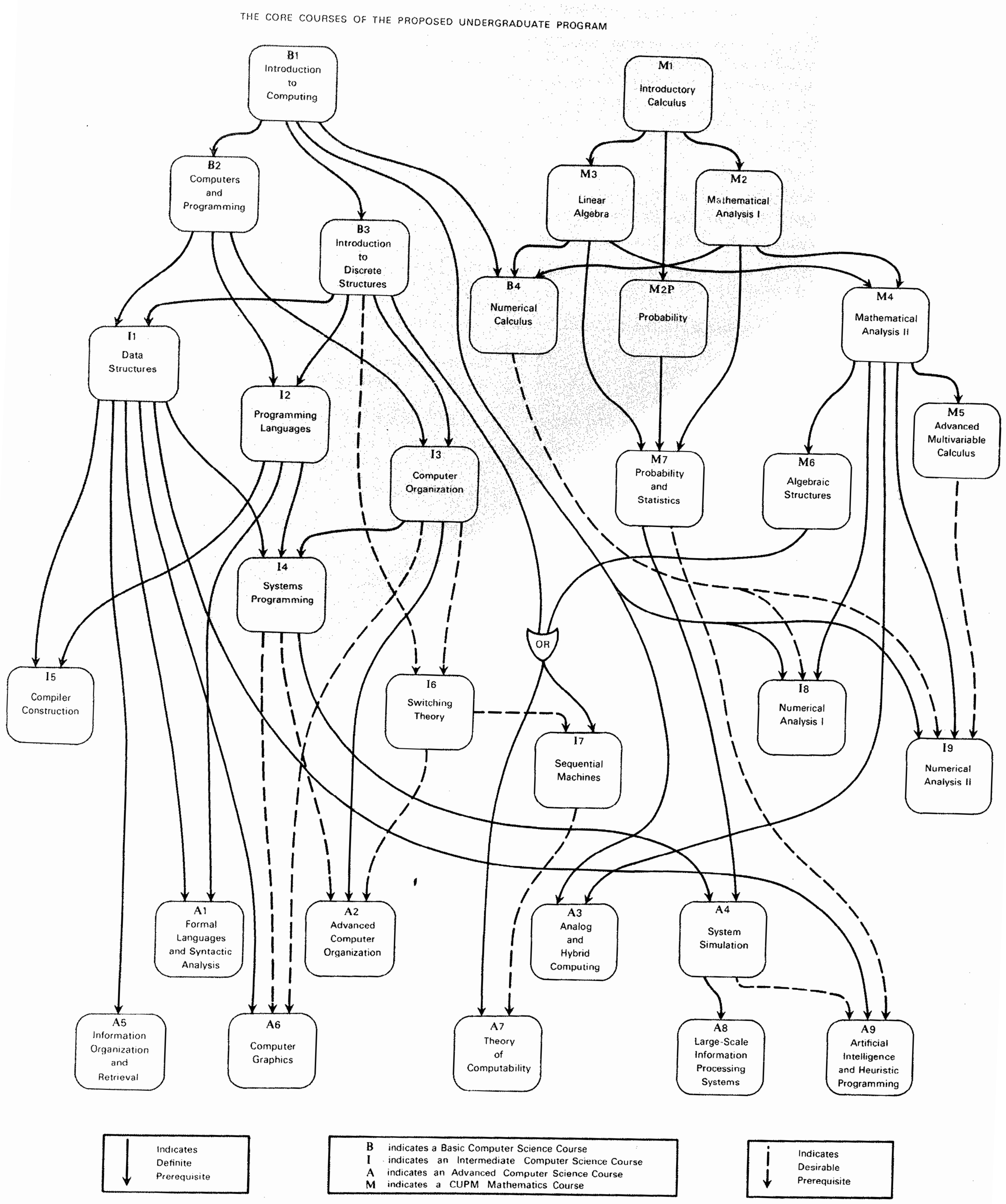
THE CORE COURSES OF THE PROPOSED UNDERGRADUATE PROGRAM
B1 Introduction to Computing
M1 Introductory Calculus
B2 Computers and Programming
B3 Introduction to Discrete Structures
M3 Linear Algebra
M2 Mathematical Analysis I
B4 Numerical Calculus
M2P Probability
M4 Mathematical Analysis II
I1 Data Structures
I2 Programming Languages
I3 Computer Organization
M5 Advanced Multivariable Calculus
M7 Probability and Statistics
M6 Algebraic Structures
I4 Systems Programming
OR
I5 Compiler Construction
I6 Switching Theory
I7 Sequential Machines
I8 Numerical Analysis I
I9 Numerical Analysis II
A1 Formal Languages and Syntactic Analysis
A2 Advanced Computer Organization
A3 Analog and Hybrid Computing
A4 System Simulation
A5 Information Organization and Retrieval
A6 Computer Graphics
A7 Theory of Computability
A8 Large-Scale Information Processing Systems
A9 Artificial Intelligence and Heuristic Programming
Indicates Definite Prerequisite
B indicates a Basic Computer Science Course
I indicates an Intermediate Computer Science Course
A indicates an Advanced Computer Science Course
M indicates a CUPM Mathematics Course
Indicates Desirable Prerequisite

FIG. 1. Prerequisite structure of courses

systems organization, logic design, micro-programming, and interpreters. Symbolic coding and assembly systems, macro definition and generation, and program segmentation and linkage. Systems and utility programs, programming techniques, and recent developments in computing. Several computer projects to illustrate basic machine structure and programming techniques.

☐This course introduces the student to those fundamental algebraic, logical, and combinatoric concepts from mathematics needed in the subsequent computer science courses and shows the applications of these concepts to various areas of computer science.

Course B3. Introduction to Discrete Structures (3-0-3)

Prerequisite: Course B1.

Review of set algebra including mappings and relations. Algebraic structures including semigroups and groups. Elements of the theory of directed and undirected graphs. Boolean algebra and propositional logic. Applications of these structures to various areas of computer science.

☐This course provides the student with an introduction to the basic numerical algorithms used in scientific computer work—thereby complementing his studies in beginning analysis—and affords him an opportunity to apply the programming techniques he has learned in Course B1. Because of these aims, many of the standard elementary numerical analysis courses now offered in mathematics departments cannot be considered as substitutes for this course.

Course B4. Numerical Calculus (2-2-3)

Prerequisites: Courses B1, M2, and M3.

An introduction to the numerical algorithms fundamental to scientific computer work. Includes elementary discussion of error, polynomial interpolation, quadrature, linear systems of equations, solution of nonlinear equations, and numerical solution of ordinary differential equations. The algorithmic approach and the efficient use of the computer are emphasized.

☐This course is concerned with one of the most fundamental—but often inadequately recognized—areas of computer science. Its purpose is to introduce the student to the relations which hold among the elements of data involved in problems, the structures of storage media and machines, the methods which are useful in representing structured data in storage, and the techniques for operating upon data structures.

Course I1. Data Structures (3-0-3)

Prerequisites: Courses B2 and B3.

Basic concepts of data. Linear lists, strings, arrays, and orthogonal lists. Representation of trees and graphs. Storage systems and structures, and storage allocation and collection. Multilinked structures. Symbol tables and searching techniques. Sorting (ordering) techniques. Formal specification of data structures, data structures in programming languages, and generalized data management systems.

☐The following intermediate course is designed to present a systematic approach to the study of programming languages and thus provide the student with the knowledge necessary to learn and evaluate such languages.

Course I2. Programming Languages (3-0-3)

Prerequisites: Courses B2 and B3.

Formal definition of programming languages including specification of syntax and semantics. Simple statements including precedence, infix, prefix, and postfix notation. Global properties of algorithmic languages including scope of declarations, storage allocation, grouping of statements, binding time of constituents, subroutines, coroutines, and tasks. List processing, string manipulation, data description, and simulation languages. Run-time representation of program and data structures.

☐The following course discusses the organization, logic design, and components of digital computing systems. It can be thought of as a continuation of the hardware concepts introduced in Course B2.

Course I3. Computer Organization (3-0-3) or (3-2-4)

Prerequisites: Courses B2 and B3.

Basic digital circuits, Boolean algebra and combinational logic, data representation and transfer, and digital arithmetic. Digital storage and accessing, control functions, input-output facilities, system organization, and reliability. Description and simulation techniques. Features needed for multiprogramming, multiprocessing, and real-time systems. Other advanced topics and alternate organizations.

☐The following course is concerned primarily with the software organization—and to a lesser extent the hardware—of computer systems which support a wide variety of users. It is intended to bring together the concepts and techniques developed in the previous courses on data structures, programming languages, and computer organization by considering their role in the design of general computer systems. The problems which arise in multiaccessing, multiprogramming, and multiprocessing are emphasized.

Course I4. Systems Programming (3-0-3)

Prerequisites: Courses I1, I2, and I3.

Review of batch process systems programs, their components, operating characteristics, user services and their limitations. Implementation techniques for parallel processing of input-output and interrupt handling. Overall structure of multiprogramming systems on multiprocessor hardware configurations. Details on addressing techniques, core management, file system design and management, system accounting, and other user-related services. Traffic control, interprocess communication, design of system modules, and interfaces. System updating, documentation, and operation.

☐The following course is intended to provide a detailed understanding of the techniques used in the design and implementation of compilers.

Course I5. Compiler Construction (3-0-3)

Prerequisites: Courses I1 and I2.

Review of program language structures, translation, loading, execution, and storage allocation. Compilation of simple expressions and statements. Organization of a compiler including compile-time and run-time symbol tables, lexical scan, syntax scan, object code generation, error diagnostics, object code optimization techniques, and overall design. Use of compiler writing languages and bootstrapping.

☐This course introduces the theoretical principles and mathematical techniques involved in the design of digital system logic. A course compatible with the content and approach of this course is frequently taught in departments of electrical engineering.

Course I6. Switching Theory (3-0-3) or (2-2-3)

Prerequisites: Courses B3 (desirable) and I3 (desirable, as it would allow more meaningful examples to be used).

Switching algebra, gate network analysis and synthesis, Boolean algebra, combinational circuit minimization, sequential circuit analysis and synthesis, sequential circuit state minimization, hazards and races, and elementary number systems and codes.

☐This theoretical course is especially recommended for undergraduate students planning to do graduate work in computer science. It is also an appropriate course for electrical engineers and may sometimes be available from or jointly developed with an electrical engineering department.

Course I7. Sequential Machines (3-0-3)

Prerequisites: Courses B3 or M6, and I6 (desirable).

Definition and representation of finite state automata and sequential machines. Equivalence of states and machines, congruence, reduced machines, and analysis and synthesis of machines. Decision problems of finite automata, partitions with the substitution property, generalized and incomplete machines, semigroups and machines, probabilistic automata, and other topics.

☐The following two courses in numerical analysis are intended to be mathematically rigorous and at the same time computer-oriented.

Course I8. Numerical Analysis I (3-0-3)

Prerequisites: Courses B1, B4 (desirable), and M4.

A thorough treatment of solutions of equations, interpolation and approximations, numerical differentiation and integration, and numerical solution of initial value problems in ordinary differential equations. Selected algorithms will be programmed for solution on computers.

Course I9. Numerical Analysis II (3-0-3)

Prerequisites: Courses B1, B4 (desirable), M4, and M5 (desirable).

The solution of linear systems by direct and iterative methods, matrix inversion, the evaluation of determinants, and the calculation of eigenvalues and eigenvectors of matrices. Application to boundary value problems in ordinary differential equations. Introduction to the numerical solution of partial differential equations. Selected algorithms will be programmed for solution on computers.

☐The following course serves as an introduction both to the theory of context-free grammars and formal languages, and to syntactic recognition techniques for recognizing languages specified by context-free grammars.

Course A1. Formal Languages and Syntactic Analysis (3-0-3)

Prerequisites: Courses I1 and I2.

Definition of formal grammars: arithmetic expressions and precedence grammars, context-free and finite-state grammars. Algorithms for syntactic analysis: recognizers, backtracking, operator precedence techniques. Semantics of grammatical constructs: reductive grammars, Floyd productions, simple syntactical compilation. Relationship between formal languages and automata.

☐The following advanced course in computer organization is centered around the comparison of solutions to basic design problems which have been incorporated in a number of quite different computers.

Course A2. Advanced Computer Organization (3-0-3)

Prerequisites: Courses I3, I4 (desirable), and I6 (desirable).

Computer system design problems such as arithmetic and nonarithmetic processing, memory utilization, storage management, addressing, control, and input-output. Comparison of specific examples of various solutions to computer system design problems. Selected topics on novel computer organizations such as those of array or cellular computers and variable structure computers.

☐This course is designed to give the computer science student some experience with analog, hybrid, and related techniques. It could also be very valuable as a service course.

Course A3. Analog and Hybrid Computing (2-2-3)

Prerequisites: Courses B1 and M4. (The CUPM mathematical analysis courses include some differential equations; more may be needed.)

Analog, hybrid and related digital techniques for the solution of differential equations. Analog simulation languages. Scaling methods. Operational characteristics of analog components. Digital differential analyzers. Analog-to-digital and digital-to-analog conversion. Stability problems. Modeling methods. Use of analog and hybrid equipment and of digital simulation of continuous systems.

☐The following course is concerned with the simulation and modeling of discrete systems on a computer. Since simulation is one of the most common applications of computers and is used to a great extent in the design of computing machines and systems, students of computer science should become acquainted with simulation techniques and their use.

Course A4. System Simulation (3-0-3)

Prerequisites: Courses I4 and M7.

Introduction to simulation and comparison with other techniques. Discrete simulation models, and introduction to, or review of, queueing theory and stochastic processes. Comparison of discrete change simulation languages. Simulation methodology including generation of random numbers and variates, design of simulation experiments for optimization, analysis of data generated by simulation experiments, and validation of simulation models and results. Selected applications of simulation.

☐The purpose of the following course is to provide an introduction to natural language processing, particularly as it relates to the design and operation of automatic information systems. Included are techniques for organizing, storing, matching, and retrieving structured information on digital computers, as well as procedures useful for the optimization of search effectiveness.

Course A5. Information Organization and Retrieval (3-0-3)

Prerequisite: Course I1.

Structure of semiformal languages and models for the representation of structured information. Aspects of natural language processing on digital computers. The analysis of information content by statistical, syntactic, and logical methods. Search and matching techniques. Automatic retrieval systems, question-answering systems. Production of secondary outputs. Evaluation of retrieval effectiveness.

☐The objective of the following course is to study the problems of handling graphic information, such as line drawings, block diagrams, handwriting, and three-dimensional surfaces, in computers. Input-output and representation-storage of pictures will be introduced from the hardware and software points of view. The course is intended to serve both the student interested in specializing in computer graphics per se and the student who seeks to apply graphic techniques to his particular computing work.

Course A6. Computer Graphics (2-2-3)

Prerequisites: Courses I1, I3 (desirable), and I4 (desirable).

Display memory, generation of points, vectors, etc. Interactive versus passive graphics. Analog storage of images on microfilm, etc. Digitizing and digital storage. Pattern recognition by features, syntax tables, random nets, etc. Data structures and graphics software. The mathematics of three-dimensions, projections, and the hidden-line problem. "Graphical programs," computer-aided design and instruction, and animated movies.

☐The following course uses abstract machines as models in the study of computability and computational complexity. Emphasis is placed on the multi-tape Turing machine as a suitable model, but other models are also considered.

Course A7. Theory of Computability (3-0-3)

Prerequisites: Courses B3 or M6, and I7 (desirable).

Introduction to Turing machines, Wang machines, Shepherdson-Sturgis, and other machines. Gödel numbering and unsolvability results, the halting problem, Post's correspondence problem, and relative uncomputability. Machines with restricted memory access, limited memory, and limited computing time. Recursive function theory and complexity classification. Models of computation including relationships to algorithms and programming.

☐The following course is intended for students who are interested in the application of information technology in large-scale information processing systems. The term "information processing system" is used here to include the hardware, software, procedures, and techniques that are assembled and organized to achieve some desired objectives. Examples of such large-scale information processing systems are business data processing systems, information storage and retrieval systems, command and control systems, and computer centers.

Course A8. Large-scale Information Processing Systems (3-0-3)

Prerequisites: Course A4, and a course in operations research or optimization theory.

Organization of major types of information processing systems. Data organization and storage structure techniques. Designing "best" systems by organizing files and segmenting problems into computer programs to make efficient use of hardware devices. Documentation methods and techniques for modifying systems. Use of optimization and simulation as design techniques. Communication problems among individuals involved in system development.

☐The following course introduces the student to those nonarithmetical applications of computing machines that: (1) attempt to achieve goals considered to require human mental capabilities (artificial intelligence); (2) model highly organized intellectual activity (simulation of cognitive behavior); and (3) describe purposeful behavior of living organisms or artifacts (self-organizing systems). Courses in this area are often taught with few prerequisites, but by requiring some or all of the prerequisites listed here this course could be taught at a more advanced level.

Course A9. Artificial Intelligence and Heuristic Programming (3-0-3)

Prerequisites: Courses I1, A4 (desirable), and M7 (desirable); and some knowledge of experimental and theoretical psychology would also be useful.

Definition of heuristic versus algorithmic methods, rationale of heuristic approach, description of cognitive processes, and approaches to mathematical invention. Objectives of work in artificial intelligence, simulation of cognitive behavior, and self-organizing systems. Heuristic programming techniques including the use of list processing languages. Survey of examples from representative application areas. The mind-brain problem and the nature of intelligence. Class and individual projects to illustrate basic concepts.

4. Undergraduate Programs

As indicated in the Introduction, there has been considerable discussion on the desirability of undergraduate degree programs in computer science. Many who favor these programs believe that an undergraduate computer science "major" is as natural today as a major in established fields such as mathematics, physics, or electrical engineering. Many who oppose these programs feel that, although undergraduate courses in computer science should be available for support of work in other areas, to offer an undergraduate degree in computer science may encourage too narrow a specialization at the expense of breadth. They point out that the lack of such breadth may be a serious handicap to a student desiring to do graduate work in computer science, and they contend that it would be better for the student to major in some established discipline while taking a number of computer science courses as supporting work. To meet these objections the Committee has made every effort to present a curriculum which includes a broad representation of basic concepts and an adequate coverage of professional techniques.

The number of undergraduate degree programs now in existence or in the planning stages—approximately one third of all Ph.D. granting institutions in the United States either have such computer science programs now or expect to have them by 1970 [5]—indicates that a discussion of the desirability of such programs is much less relevant than the early development of guidelines and standards for them. The Committee feels strongly, however, that schools should exercise caution against the premature establishment of undergraduate degree programs. The pressures created by large numbers of students needing to take courses required for the degree could easily result in a general lowering of standards exactly when it is vital that such programs be established and maintained only with high standards.

The variation of undergraduate program requirements among and within schools dictates that this Committee's recommendations must be very general. It is fully expected that each individual school will modify these recommendations to meet its specific circumstances, but it is hoped that these modifications will be expansions of or changes in the emphasis of the basic program proposed, rather than reductions in quantity or quality. The requirements recommended herein have been kept to a minimum in order to allow the student to obtain a "liberal education" and to enable individual programs to add additional detailed requirements. Since the liberal education requirements of each school are already well established, the Committee has not considered making recommendations on such requirements.

The Committee's recommendations for an undergraduate computer science curriculum are stated in terms of computer science course work, programming experience, mathematics course work, technical electives, and possible areas of specialization. Some suggestions are also given as to how the courses might fit chronologically into a semester-by-semester schedule.

Computer Science Courses. The basic and intermediate course requirements listed below emphasize the first two "major subject divisions"—namely, "information structures and processes" and "information processing systems"—described in Section 2 of this report. These courses should give the student a firm grounding in the fundamentals of computer science:

The major in computer science should consist of at least 30 semester hours including the courses:

B1. Introduction to Computing
B2. Computers and Programming
B3. Introduction to Discrete Structures
B4. Numerical Calculus
I1. Data Structures
I2. Programming Languages
I3. Computer Organization
I4. Systems Programming

and at least two of the courses:

I5. Compiler Construction
I6. Switching Theory
I7. Sequential Machines
I8. Numerical Analysis I
I9. Numerical Analysis II

Programming Experience. Developing programming skill is by no means the main purpose of an undergraduate program in computer science; nevertheless, such skill is an important by-product. Therefore such a program should insure that the student attains a reasonable level of programming competence. This can be done in part by including computer work of progressive complexity and diversity in the required courses, but it is also desirable that each student participate in a "true-to-life" programming project. This might be arranged through summer employment, a cooperative work-study program, part-time employment in computer centers, special project courses, or some other appropriate means.

Mathematics Courses. The Committee feels that an academic program in computer science must be well based in mathematics since computer science draws so heavily upon mathematical ideas and methods. The recommendations for required mathematics courses given below should be regarded as minimal; obviously additional course work in mathematics would be essential for students specializing in numerical applications.

The supporting work in mathematics should consist of at least 18 hours including the courses:

- M1. Introductory Calculus
- M2. Mathematical Analysis I
- M2P. Probability
- M3. Linear Algebra

and at least two of the courses:

- M4. Mathematical Analysis II
- M5. Advanced Multivariate Calculus
- M6. Algebraic Structures
- M7. Probability and Statistics

Technical Electives. Assuming that a typical four-year curriculum consists of 124 semester hours, a number of technical electives beyond the requirements listed above should be available to a student in a computer science program. Some of these electives might be specified by the program to develop a particular orientation or minor. Because of the temptation for the student to overspecialize, it is suggested that a limit be placed on the number of computer science electives a student is allowed to take—for example, three such courses might be permitted. For many students it will be desirable to use the remaining technical electives to acquire a deeper knowledge of mathematics, physical science, electrical engineering, or some other computer-related field.

Students should be carefully advised in the choice of their electives. In particular, those preparing for graduate school must insure that they will be qualified for admission into the program of their choice. Those seeking a more "professional" education can specialize to some extent through the proper choice of electives.

Areas of Specialization. Although undue specialization is not appropriate at the undergraduate level, the technical electives may be used to orient undergraduate programs in a number of different directions. Some of the possible orientations, along with appropriate courses (of Section 3) and subject areas (of Section 2) from which the optional and elective courses might be taken, are given below.

APPLIED SYSTEMS PROGRAMMING

Optional courses

- I5. Compiler Construction
- I6. Switching Theory

Electives from courses

- A2. Advanced Computer Organization
- A5. Information Organization and Retrieval
- A6. Computer Graphics

Electives from areas

- IV.8 Combinatorial Mathematics
- IV.9 Mathematical Logic
- IV.11 Probability and Statistics
- IV.12 Operations Analysis

COMPUTER ORGANIZATION AND DESIGN

Optional courses

- I6. Switching Theory
- I7. Sequential Machines

Electives from courses

- A2. Advanced Computer Organization
- A4. System Simulation
- A8. Large-scale Information Processing Systems

Electives from areas

- IV.3 Differential Equations
- V.2 Basic Electronics
- V.6 Digital and Pulse Circuits
- V.7 Coding and Information Theory

SCIENTIFIC APPLICATIONS PROGRAMMING

Optional courses

- I8. Numerical Analysis I
- I9. Numerical Analysis II

Electives from courses

- A3. Analog and Hybrid Computing
- A4. System Simulation
- A5. Information Organization and Retrieval
- A6. Computer Graphics

Electives from areas

- IV.3 Differential Equations
- IV.7 Optimization Theory
- V.4 Thermodynamics and Statistical Mechanics
- V.5 Field Theory

DATA PROCESSING APPLICATIONS PROGRAMMING

Optional courses

- I5. Compiler Construction
- I6. Switching Theory

Electives from courses

- A4. System Simulation
- A5. Information Organization and Retrieval
- A8. Large-scale Information Processing Systems

Electives from areas

- IV.7 Optimization Theory
- IV.11 Probability and Statistics
- IV.12 Operation Analysis
- V.7 Coding and Information Theory

Semester Chronology. Any institution planning an undergraduate program based on the recommendations of this report should work out several complete four-year curricula to insure that the required courses mesh in an orderly manner with electives and with the "general education" requirements of the institution. This will help the school to take into account local circumstances such as having very few entering freshmen who can begin college mathematics with the calculus.

Table I gives some examples of how a student in computer science might be scheduled for the minimum set of courses recommended for all majors.

TABLE I

Year	*Semester*	*First example*	*Second example*	*Third example*
Freshman	First	M1, B1	Basic Math.	Basic Math.
	Second	M2, B2	M1, B1	Basic Math.
Sophomore	First	M3, B3	M2, B2	M1, B1
	Second	M4, B4	M3, B3	M2, B2
Junior	First	M2P, I1	M4, B4	M2P, B3, B4
	Second	M5, I2	M2P, I1	M3, I1, I2
Senior	First	I3, I8	M7, I2, I3	M6, I3, I6
	Second	I4, I9	I4, I5, I6	M7, I4, I7

5. Master's Degree Programs

The recommendations given in this section concern undergraduate preparation for graduate study in computer science, requirements for a Master of Science degree in computer science, and some possible areas of concentration for students who are at the master's degree level.

Undergraduate Preparation. The recommended preparation for graduate study in computer science consists of three parts as listed below. The course work which would provide this background is indicated in parentheses.

a. Knowledge of computer science including algorithmic processes, programming, computer organization, discrete structures, and numerical mathematics. (Courses B1, B2, B3, and B4 or I8 of Section 3.)

b. Knowledge of mathematics, including the calculus and linear algebra, and knowledge of probability and statistics. (Courses M1, M2, M3, M4, M2P, M7 of CUPM.)

c. Additional knowledge of some field such as computer science, mathematics, electrical engineering, physical science, biological science, linguistics, library science, or management science which will contribute to the student's graduate study in computer science. (Four appropriate courses on an intermediate level.)

A student with a bachelor's degree in computer science, such as recommended in Section 4, can have taken all these prerequisites as basic and supporting courses and can also have taken further work which overlaps with some of the subject matter to be treated at the master's level. Although such a student will be able to take more advanced graduate work in computer science to satisfy his master's requirements, he may need to take more supporting work than a student whose undergraduate degree was in some other field. A student with an undergraduate degree in mathematics, physical science, or electrical engineering can easily qualify for such a program if he has taken adequate supporting courses in computer science. Other applicants should have no more than a few deficiencies in order to qualify.

In the near future many of the potential students, because of having completed their undergraduate work some time ago, will not have had the opportunity to meet these requirements. Liberal policies should therefore be established so that promising students can make up deficiencies.

Degree Requirements. Each student's program of study for the master's degree should have both breadth and depth. In order to obtain breadth, the student should take course work from each of the three subject divisions of computer science described in Section 2. To obtain depth, he should develop an area of concentration in which he would write a master's thesis or complete a master's project (if required).

The master's degree program in computer science should consist of at least nine courses. Normally at least two courses—each in a different subject area—should be taken from each of the following subject divisions of computer science:

I. Information Structures and Processes
II. Information Processing Systems
III. Methodologies

Sufficient other courses in computer science or related areas should be taken to bring the student to the forefront of some area of computer science.

In order that the student may perform in his required course work at the graduate level he must acquire a knowledge of related areas, such as mathematics and the physical sciences, either as part of his undergraduate preparation or as part of his graduate program. Computer science as a discipline requires an understanding of mathematical methods and an ability to use mathematical techniques beyond the specific undergraduate preparation in mathematics recommended above. Hence, a student who does not have a "strong" mathematics background should take either further courses in mathematics, or he should take computer science courses which contain a high mathematical content.

If Courses I1, I2, I3, and I4 of Section 3 are taught at a sufficiently high level, they can be used to satisfy the "breadth" requirements for the first two subject divisions listed above. In any case, the student who has taken such courses as part of his undergraduate program could take more advanced courses in these areas so that the requirement for two courses in each subject division might be relaxed somewhat. This might permit such a student to take more supporting work outside computer science.

Areas of Concentration. The "depth" requirement will often involve courses from fields other than computer science, so that a student may have to take additional courses in these fields just to meet prerequisites unless he has anticipated this need in his undergraduate preparation. In any event, the particular courses a student selects from each of the three subject divisions of computer science should be coordinated with his area of concentration. To illustrate how this might be done, six possible concentrations are shown below together with lists of the subject areas (of Section 2) from which appropriate courses might be selected for each of the concentrations. The characterization of courses in terms of subject areas instead of explicit content effectively gives a list of suggested topics which can be drawn upon in designing master's level courses suited to the needs of individual institutions.

THEORETICAL COMPUTER SCIENCE

- I.1 Data Structures
- I.2 Programming Languages
- I.3 Models of Computation
- III.3 Symbol Manipulation
- III.8 Artificial Intelligence
- IV.8 Combinatorial Analysis
- IV.9 Mathematical Logic
- V.7 Coding and Information Theory

APPLIED SOFTWARE

- I.1 Data Structures
- I.2 Programming Languages
- II.1 Computer Design and Organization
- II.2 Translators and Interpreters
- II.3 Computer and Operating Systems
- III.3 Symbol Manipulation
- III.6 Simulation
- IV.7 Optimization Theory
- IV.9 Mathematical Logic

APPLIED HARDWARE

- I.1 Data Structures
- I.3 Models of Computation
- II.1 Computer Design and Organization
- II.3 Computer and Operating Systems
- III.5 Computer Graphics
- IV. 7 Optimization Theory
- IV. 9 Mathematical Logic
- V.6 Digital and Pulse Circuits
- V.7 Coding and Information Theory

NUMERICAL MATHEMATICS

- I.1 Data Structures
- I.2 Programming Languages
- II.1 Computer Design and Organization
- II.3 Computer and Operating Systems
- III.1 Numerical Mathematics
- III.6 Simulation
- IV.5 Theoretical Numerical Analysis
- IV.6 Methods of Applied Mathematics
- IV.7 Optimization Theory

INSTRUMENTATION

- I.1 Data Structures
- I.2 Programming Languages
- II.1 Computer Design and Organization
- II.4 Special Purpose Systems
- III.6 Simulation
- III.9 Process Control
- IV.6 Methods of Applied Mathematics
- IV.7 Optimization Theory
- V.8 Communication and Control Theory

INFORMATION SYSTEMS

- I.1 Data Structures
- I.2 Programming Languages
- II.1 Computer Design and Organization
- II.3 Computer and Operating Systems
- III.2 Data Processing and File Management
- III.4 Text Processing
- III. 7 Information Retrieval
- IV.7 Optimization Theory
- IV.9 Mathematical Logic

The requirement of a master's thesis or other project has been left unspecified since general institutional requirements will usually determine this. It is strongly recommended, however, that a master's program in computer science contain some formal provision for insuring that the student gains or has gained project experience in computer applications. This could be effected by requiring that students carry out either individually or cooperatively a substantial assigned task involving analysis and programming, or better, that students be involved in an actual project on campus or in conjunction with other employment.

This proposed program embodies sufficient flexibility to fulfill the requirements of either an "academic" degree obtained in preparation for further graduate study or a terminal "professional" degree. Until clearer standards both for computer science research and the computing profession have emerged, it seems unwise to attempt to distinguish more definitely between these two aspects of master's degree programs.

6. Doctoral Programs

Academic programs at the doctoral level reflect the specific interests of the faculty and, hence, vary from university to university. Therefore, the Committee cannot expect to give recommendations for such doctoral programs in as great a detail as has been done for the undergraduate and master's degree programs. The large number of institutions planning such programs and the variety of auspices under which they are being sponsored, however, suggest that a need exists for guidelines as to what constitutes a "good" doctoral program. While recommendations on doctoral programs will not be given at this time, the problem of how to obtain such guidelines has been of considerable interest to the Committee.

One possible source of such guidelines is the existing doctoral programs. A description of the program at Stanford University [15] has already been published in *Communications of the ACM* and descriptions of many other programs are available from the universities concerned. Information based on a number of such programs is contained in the report of the June 1967, Stony Brook Conference [11]. This report also contains a list of thesis topics currently being pursued or recently completed. In the future the Curriculum Committee hopes to encourage wide dissemination of the descriptions of existing programs and research topics. Perhaps it can take an active role in coordinating the interchange of such information.

In 1966 Professor Thomas Hull was asked by ACM to examine the question of doctoral programs in computer science. After discussion with the members of this Committee and with many other interested persons, Professor Hull decided to solicit a series of articles on the research and teaching areas which might be involved in doctoral programs. Each article is to be written by an expert in the particular subject area, such as programming languages, systems programming, computer organization, numerical mathematics, automata theory, large systems, and artificial intelligence. Each article is to attempt to consider all aspects of the subject area which might be helpful to those developing a graduate program, including as many of the following topics as possible:

a. Definition of the subject area, possibly in terms of an annotated bibliography.
b. Prerequisites for work in the area at the doctoral level.
c. Outlines of appropriate graduate courses in the area.
d. Examples of questions for qualifying examinations in the area.
e. Indication of suitable thesis topics and promising directions for research in the area.
f. The extent to which the subject area ought to be required of all doctoral students in computer science.

These articles are scheduled for publication in *Communications of the ACM* and it is hoped that they will stimulate further articles on doctoral programs.

7. Service Courses, Minors, and Continuing Education

Though it is now generally recognized that a significant portion of our undergraduate students needs some knowledge of computing, the amount and type of computing knowledge necessary for particular areas of study are still subject to considerable discussion. The Pierce Report [7] estimates that about 75 percent of all college undergraduates are enrolled in curricula where some computer training would be useful. This estimate, based on figures compiled by the US Office of Education, involves dividing the undergraduate student population into three groups. The first group, about 35 percent of all undergraduates, consists of those in scientific or professional programs having a substantial quantitative content (e.g. mathematics, physics, and engineering). At least some introductory knowledge of computing is already considered highly desirable for almost all of these students. The second group, some 40 percent, is made up of those majoring in fields where an understanding of the fundamentals of computing is steadily becoming more valuable (e.g. business, behavioral sciences, education, medicine, and library science). Many programs in these areas are already requiring courses in computing, and most are expected to add such requirements in the future. The third group, roughly 25 percent, comprises those undergraduates who are majoring in areas which do not necessarily depend on the use of computers (e.g. music, drama, literature, foreign languages, liberal arts, and fine arts). There are many persons who maintain that even these students could benefit from a course which would give them an appreciation of this modern technology and its influence on the structure of our society.

The extent and nature of the courses on computing needed for these three groups of students should be given further careful study, but the existence of a substantial need for service courses in computer science seems undeniable. Students in the more quantitative fields are usually well-equipped to take the basic courses designed for the computer science major. In particular, Course B1 should serve as an excellent introductory course for these students and, depending upon their interests, Course B2 or B4 might serve as a second course. When these students develop a greater interest in computing, they should normally be able to select an appropriate "minor" program of study from the courses described in Section 3. In developing a minor program careful consideration should be given to the comparative values of each course in the development of the individual student.

Some special provisions appear to be necessary for students in the second and third groups described above. A special version of Course B1 which would place more emphasis on text processing and other nonnumeric applications might be more appropriate for students in the second group. However, it is important that this course provides adequate preparation for such courses as B2 and B3, since many of these students might be expected to take further courses in computer science. It may also be desirable to develop courses giving primary emphasis to the economic, political, sociological, and other implications of the growing use of computer technology. Such courses would not be considered substitutes for basic technical courses such as B1, but they could serve the needs of the third group of students.

Professional programs at all levels offer limited opportunity for courses outside their highly structured curricula, and they also present special problems. In some cases it may be necessary to develop special courses for students in such programs or to integrate work on computing into existing courses. Those preparing for graduate professional programs will often find it desirable to include some of the basic computer science courses in their undergraduate work.

The responsibility for developing and conducting the basic service courses in computer science should be concentrated within the academic structure and combined with the operation of educational programs in computer science. By properly aggregating students from similar fields, those responsible for planning academic courses can make them more generally applicable and broadly directed. Under this arrangement teachers can be used more effectively and course content can more easily be kept current with the rapidly moving developments in the field. Also, students who find a need or a desire to delve further into computer science are more likely to have the necessary background to take advanced courses. On the other hand, it must be recognized that some departments will have many situations where special applications of the computer can best be introduced in their own courses. Certainly those responsible for the basic computer science service courses must be sensitive to the needs of the students for whom these courses are intended.

Finally, the need for continuing education in computer science must be recognized. Much of the course material discussed in this report did not exist ten or fifteen years ago, and practically none of this material was available to students until the last few years. Anyone who graduated from college in the early 1960's and whose "major" field of study is related to computing is already out-of-date unless he has made a determined effort to continue his education. Those responsible for academic programs in computer science and those agencies which help to direct and support continuing education should be especially alert to these needs in this unusually dynamic and important field.

8. Implementation

In educational institutions careful consideration should be given to the problems of implementing a computer-related course of study—be it a few introductory or service courses, an undergraduate degree program, or a graduate degree program. Some of these problems involve organization, staff requirements, and physical facilities (including computing services). Although individual ways of providing a favorable environment for computer science will be found in each school, the following discussion is intended to call attention to the extent of some of these problems.

Organization for Academic Programs. It should be realized that the demands for education in computer science are strong. If some suitable place in the institutional structure is not provided for courses and programs in computer science to be developed, they will spring up within a number of existing departments and a possible diffusion of effort will result as has been experienced with statistics in many universities.

If degree programs in computer science are to be offered, it is desirable to establish an independent academic unit to administer them. Such a unit is needed to provide the appropriate mechanisms for faculty appointments and promotions, for attention to continuing curriculum development, and for the allocation of resources such as personnel, budget, space, and equipment. This academic unit should also be prepared to provide general service courses and to cooperate in developing computer-oriented course work in other departments and professional schools.

Many universities have established departments of computer science as part of their colleges of arts and sciences, and some have established divisions of mathematical sciences, which include such departments as mathematics, applied mathematics, statistics, and computer science. Other institutions have located computer science departments in colleges of engineering and applied science. Academic units in computer science have also been affiliated with a graduate school, associated with more than one college, or even established independent of any college in a university.

The organizational problems for this new field are serious, and their solution will inevitably require new budget commitments from a university. However, failure to come to grips with the problem will probably prove more costly in the long run; duplicated courses and programs of diluted quality may result, and a major upheaval may eventually be required for reorganization.

Staff Requirements. Degree programs in computer science require a faculty dedicated to this discipline—that is, individuals who consider themselves computer scientists regardless of their previous academic training. Although graduate programs in computer science are now producing a limited number of potential faculty members, the demand for such people in industry and government and the competition for faculty among universities are quite intense. Hence educational institutions will have to obtain most of their computer science faculty from other sources—at least for the immediate future. Many faculty members in other departments of our universities have become involved with computing and have contributed to its development to the point that they are anxious to become part of a computer science program. Within industry and government there are also people with the necessary academic credentials who are willing to teach the technology they have helped develop. Thus, extensive experience and academic work in computer science, accompanied by academic credentials in a related area such as mathematics, electrical engineering, or other appropriate disciplines, can serve as suitable qualifications for staff appointments in a computer science program. Joint appointments with other academic departments or with the computing center can help fill some of the need, but it is desirable that a substantial portion of the faculty be fully committed to computer science. Moreover, there is some critical size of faculty—perhaps the equivalent of five full-time positions—which is needed to provide a reasonable coverage of the areas of computer science discussed in Section 2.

Since relatively few good textbooks are available in the computer sciences, the computer science faculty will need to devote an unusually large part of its time to searching the literature and developing instructional materials. This fact should be taken into consideration in determining teaching loads and staff assignments.

Physical Facilities. Insofar as physical facilities are concerned, computer science should generally be included among the laboratory sciences. Individual faculty members may need extra space to set up and use equipment, to file cards and voluminous computer listings, and otherwise to carry out their teaching and research. In addition to normal library facilities, special collections of material, such as research reports, computer manuals, and computer programs, must be obtained and facilities made available for their proper storage and effective use. Space must also be provided for keypunches, remote consoles to computers, and any other special equipment needed for education and/or research. Laboratory-type classrooms must be available to allow students, either as individuals or as groups, to spread out and study computer listings.

It is no more conceivable that computer science courses—let alone degree programs—can exist without a computer available to students than that chemistry and physics offerings can exist without the associated laboratory equipment. Degree programs require regular access to at least a medium-sized computer system of sufficient complexity in configuration to require the use of an operating system. The total operating costs of such systems are at least $20,000 per month. In terms of hours per month, the machine requirements of computer science degree programs will vary according to the number of students enrolled, the speed of the computer and the efficiency of its software, and the philosophy of the instructors. It is entirely possible that an undergraduate degree program might require as much as four hours of computing on a medium-sized computer per class day.

Space for data and program preparation and program checking must be provided, and the logistics of handling hundreds and possibly thousands of student programs per day must be worked out so that each student has frequent access to the computer with a minimum of waiting and confusion. Although many of these same facilities must be provided for students and faculty other than those in computer science, the computer science program is particularly dependent on these services. It is simply false economy to hamper the use of expensive computing equipment by crowding it into unsuitable space or in some other way making it inaccessible.

The study and development of systems programs will require special forms of access to at least medium-scale computing systems. This will place an additional burden on the computer center and may possibly require the acquisition of completely separate equipment for educational and research purposes. In advanced programs it is likely that other specialized equipment will be necessary to handle such areas as computer graphics, numeric control of machines, process control, simulation, information retrieval systems, and computer-assisted instruction.

Although assistance in financing computer services and equipment can be obtained from industry and from federal and state governments, the Committee feels that universities should provide for the costs of equipment and services for computer science programs just as they provide for costs of other laboratory sciences. Based on its knowledge of costs at a number of schools the Committee estimates that computer batch processing of student jobs for elementary courses presently costs an average of about $30 per semester hour per student, whereas the Pierce Report [7] estimates that it costs colleges about $95 per chemistry student per year for a single chemistry laboratory course. Although computer costs are decreasing relative to capacity, it is expected that students will be able to use more computer time effectively in the future as computers become more accessible through the use of such techniques as time-sharing. On the basis of these estimates and expectations, future computer costs for academic programs may well approach faculty salary costs.

Relation of the Academic Program to the Computing Center. As indicated above, the demands which an academic program in computer science places on a university computing center are more than routine. Computer and programming systems must be expanded and modified to meet the growing and varied needs of these programs as well as the needs of the other users. The service function of a computer center must therefore be enhanced by an activity which might be described as "applied computer science." In a complementary way, it is appropriate for a computer science faculty to be deeply involved in the application of computers, particularly in the development of programming systems. For these reasons, the activities of a computer center and a computer science department should be closely coordinated. The sharing of staff through joint appointments helps facilitate such cooperation, and it is almost necessary to provide such academic appointments in order to attract and retain certain essential computer center personnel.

It should be realized, however, that the basic philosophies of providing services and of pursuing academic ends differ to such an extent that conflicts for attention may occur. At one extreme, the research of a computer science faculty may so dominate the activities of a computer center that its service to the academic community deteriorates. At the other extreme, the routine service demands of a computer center may inhibit the faculty's ability to do their own research, or the service orientation of a center may cause the educational program to consist of mere training in techniques having only transient value. Considerable and constant care must be taken to maintain a balance between these extremes.

REFERENCES

1. Association for Computing Machinery, Curriculum Committee on Computer Science. An undergraduate program in computer science—preliminary recommendations. *Comm. ACM 8*, 9 (Sept. 1965), 543–552.
2. Newell, A., Perlis, A. J., and Simon, H. A. Computer science. (Letter to the Editor). *Science 157*, 3795 (22 Sept. 1967), 1373–1374.
3. University of Chicago. Graduate programs in the divisions, announcements 1967–1968. U. of Chicago, Chicago, pp. 167–169.
4. Gorn, S. The computer and information sciences: a new basic discipline. *SIAM Review 5*, 2 (Apr. 1963), 150–155.
5. Hamblen, J. W. Computers in higher education: expenditures, sources of funds, and utilization for research and instruction 1964–65, with projections for 1968–69. (A report on a survey supported by NSF). Southern Regional Education Board, Atlanta, Ga., 1967.
6. Rosser, J. B., et al. Digital computer needs in universities and colleges. Publ. 1233, National Academy of Sciences-National Research Council, Washington, D. C., 1966.
7. President's Science Advisory Committee. Computers in higher education. The White House, Washington, D. C., Feb. 1967.
8. Mathematical Association of America, Committee on the Undergraduate Program in Computer Science (CUPM). Recommendations on the undergraduate mathematics program for work in computing. CUPM, Berkeley, Calif., May 1964.
9. Commission on Engineering Education, COSINE Committee. Computer sciences in electrical engineering. Commission in Engineering Education, Washington, D. C., Sept. 1967.
10. British Computer Society, Education Committee. Annual education review. *Comput. Bull. 11*, 1 (June 1967), 3–73.
11. Finerman, A. (Ed.) *University Education in Computing Science.* (Proceedings of the Graduate Academic Conference in Computing Science, Stony Brook, New York, June 5–8, 1967) ACM Monograph, Academic Press, New York, 1968.
12. Mathematical Association of America, Committee on the Undergraduate Program in Mathematics (CUPM). A general curriculum in mathematics for colleges. CUPM, Berkeley, Calif., 1965.
13. ——. Recommendations in the undergraduate mathematics program for engineers and physicists. CUPM, Berkeley, Calif., 1967.
14. ——. A curriculum in applied mathematics. CUPM, Berkeley, Calif., 1966.
15. Forsythe, G. E. A university's education program in computer science. *Comm. ACM 10*, 1 (Jan. 1967), 3–11.

Acknowledgments

The following people have served as consultants to the Committee on one or more occasions or have given considerable other assistance to our work.

Richard V. Andree, University of Oklahoma
Robert L. Ashenhurst, University of Chicago
Bruce H. Barnes, Pennsylvania State University
Robert S. Barton, University of Utah
J. Richard Buchi, Purdue University
Harry Cantrell, General Electric Company
Mary D'Imperio, Department of Defense
Arthur Evans, Massachusetts Institute of Technology
David C. Evans, University of Utah
Nicholas V. Findler, State University of New York at Buffalo
Patrick C. Fischer, University of British Columbia
George E. Forsythe, Stanford University
Bernard A. Galler, University of Michigan
Saul Gorn, University of Pennsylvania
Preston C. Hammer, Pennsylvania State University
Richard W. Hamming, Bell Telephone Laboratories
Harry D. Huskey, University of California at Berkeley
Peter Z. Ingerman, Radio Corporation of America
Donald E. Knuth, California Institute of Technology
Robert R. Korfhage, Purdue University
Donald J. Laird, Pennsylvania State University
George E. Lindamood, University of Maryland
William C. Lynch, Case Institute of Technology
M. Douglas McIlroy, Bell Telephone Laboratories
Robert McNaughton, Rensselaer Polytechnic Institute
Michel Melkanoff, University of California at Los Angeles
William F. Miller, Stanford University
Anthony G. Oettinger, Harvard University
Elliott I. Organick, University of Houston
Robert H. Owens, University of Virginia
Charles P. Reed, Jr., Georgia Institute of Technology
Saul Rosen, Purdue University
Daniel Teichroew, Case Institute of Technology
Andries van Dam, Brown University
Robert J. Walker, Cornell University
Peter Wegner, Cornell University

Written comments on the Committee's work, contributions to course outlines, and other assistance have been rendered by the following:

Bruce W. Arden, University of Michigan
John E. Bakken, Midwest Oil Corporation
Larry L. Bell, Auburn University
Robert D. Brennan, International Business Machines Corp.
Yaohan Chu, University of Maryland
Charles H. Davidson, University of Wisconsin
Harold P. Edmundson, University of Maryland
Charles W. Gear, University of Illinois
Robert T. Gregory, University of Texas
Keith Hastings, University of Toronto
Carl F. Kossack, University of Georgia
Ralph E. Lee, University of Missouri at Rolla
George Mealy, Massachusetts Institute of Technology
Harlan D. Mills, International Business Machines Corp.
Jack Minker, University of Maryland and Auerbach Corp.
Jack Noland, General Electric Company
James C. Owings, Jr., University of Maryland
David L. Parnas, Carnegie-Mellon University
Charles R. Pearson, J. P. Stevens and Co.
Tad Pinkerton, University of Michigan
Roland L. Porter, Los Angeles, California
Anthony Ralston, State University of New York at Buffalo
Roy F. Reeves, Ohio State University
John R. Rice, Purdue University
Gerard Salton, Cornell University
Gordon Sherman, University of Tennessee
Vladimir Slamecka, Georgia Institute of Technology
Joseph F. Traub, Bell Telephone Laboratories

Numerous other people have contributed to the work of the Committee through informal discussions and other means. The Committee is grateful for all of the assistance it has received and especially for the cooperative spirit in which it has been given.

Appendix. Course Outlines and Bibliographies

For each of the twenty-two courses described in Section 3, this Appendix contains a brief discussion of the approach to teaching the course, a detailed outline of the content of the course, and a bibliography listing material which should be useful to the teacher and/or the student in the course. The amount of attention which might be devoted to the various topics in the content of some of the courses is indicated by percentage or by number of lectures. Whenever possible, each bibliographic entry is followed by a reference to its review in *Computing Reviews*. The format used for these references is CR-xyvi-n, where xy indicates the year of the review, v the volume number, i the issue number, and n the number of the review itself. Most of the bibliographic entries are followed by a brief annotation which is intended to indicate the way in which the item would be useful and perhaps to clarify the subject of the item. In some cases the title is sufficient for this purpose, and no annotation is given. In other cases the items are simply keyed in various ways to the sections of the content to which they apply. Although an effort has been made to cite a wide variety of texts and reference materials for each course, space and other considerations have prevented the listing of all books and papers which might bear on the topics treated.

Course B1. Introduction to Computing (2-2-3)

APPROACH

This first course in computing concentrates on the solution of computational problems through the introduction and use of an algorithmic language. A single such language should be used for most of the course so that the students may master it well enough to attack substantial problems. It may be desirable, however, to use a simple second language of quite different character for a problem or two in order to demonstrate the wide diversity of the computer languages available. Because of its elegance and novelty, SNOBOL can be used quite effectively for this purpose. In any case, it is essential that the student be aware that the computers and languages he is learning about are only particular instances of a widespread species.

The notion of an algorithm should be stressed throughout the course and clearly distinguished from that of a program. The language structures should be carefully motivated and precisely defined using one or more of the formal techniques available. Every effort should be made to develop the student's ability to analyze complex problems and formulate algorithms for their solution. Numerous problems should be assigned for computer solution, beginning early in the course with several small projects to aid the student in learning to program, and should include at least one major project, possibly of the student's own choosing. Careful verification of program operation and clear program documentation should be emphasized.

CONTENT

This outline reflects an order in which the material might be presented; however, the order of presentation will be governed by the choice of languages and texts as well as individual preferences. In particular, the treatment of some of the topics listed below might be distributed throughout the course. Although not specifically listed in the following outline, programming and computer projects should constitute an important part of the content of this course.

1. *Algorithms, Programs, and Computers.* The concept and properties of algorithms. Flowcharts of algorithms and the need for precise languages to express algorithms. The concept of a program, examples of simple programs, and description of how computers execute programs. Programming languages including the description of their syntax and semantics. (10%)

2. *Basic Programming.* Constants, identifiers, variables, subscripts, operations, functions, and expressions. Declarations, substitution statements, input-output statements, conditional statements, iteration statements, and complete programs. (10%)

3. *Program Structure.* Procedures, functions, subroutine calling, and formal-actual parameter association. Statement grouping, nested structure of expressions and statements, local versus global variables, run-time representation, and storage allocation. Common data, segmenting, and other structural features. (10%)

4. *Programming and Computing Systems.* Compilers, libraries, loaders, system programs, operating systems, and other information necessary for the student to interact with the computer being used. (5%)

5. *Debugging and Verification of Programs.* Error conditions and messages, techniques of debugging, selection of test data, checking of computer output, and programming to guard against errors in data. (5%)

6. *Data Representation.* Systems of enumeration and binary codes. Representation of characters, fixed and floating-point numbers, vectors, strings, tables, matrices, arrays, and other data structures. (10%)

7. *Other Programming Topics.* Formatted input and output. Accuracy, truncation, and round-off errors. Considerations of efficiency. Other features of language(s) being considered. (10%)

8. *Organization and Characteristics of Computers.* Internal organization including input-output, memory-storage, processing and control. Registers, arithmetic, instruction codes, execution of instruction, addressing, and flow of control. Speed, cost and characteristics of various operations and components. (10%)

9. *Analysis of Numerical and Nonnumerical Problems.* Applications of algorithm development and programming to the solution of a variety of problems (distributed throughout the course). (15%)

10. *Survey of Computers, Languages, Systems, and Applications.* The historical development of computers, languages, and systems including recent novel applications of computers, and new developments in the computing field. (10%)

11. *Examinations.* (5%)

ANNOTATED BIBLIOGRAPHY

In addition to the materials listed here, there are numerous books and manuals on specific computer languages which would be appropriate as part of the textual material for this course. Very few books, however, place sufficient emphasis on algorithms and provide the general introductory material proposed for this course.

1. ARDEN, B. W. *An Introduction to Digital Computing.* Addison-Wesley, Reading, Mass., 1963, 389 pp. CR-6345-4551.

This text uses MAD and emphasizes the solution of numerical problems, although other types of problems are discussed. Numerous examples and exercises.

2. FORTE, A. *SNOBOL3 Primer.* M.I.T. Press, Cambridge, Mass., 1967, 107 pp.

   An elementary exposition of SNOBOL3 which might well be used to introduce a "second" language. Many exercises and examples. (SNOBOL4 is now becoming available.)

3. GALLER, B. A. *The Language of Computers.* McGraw-Hill, New York, 1962, 244 pp. CR-6341-3574.

   Emphasizes "discovering" the structure of algorithms needed for the solution of a varied set of problems. The computer language features necessary to express these algorithms are carefully motivated. The language introduced is primarily based on MAD, but FORTRAN and ALGOL are also discussed.

4. GRUENBERGER, F. The teaching of computing (Guest editorial). *Comm. ACM 8*, 6 (June 1965), 348 and 410. CR-6565-8074.

   Conveys eloquently the philosophy which should be used in developing and teaching an introductory computing course.

5. GRUENBERGER, F. AND JAFFRAY, G. *Problems for Computer Solution.* Wiley, New York, 1965, 401 pp. CR-6671-8757.

   Contains a collection of problems appropriate for computer solution by students. Student is guided into the analysis of the problems and the development of good computational solutions, but actual computer programs for the solutions are not given.

6. HULL, T. E. *Introduction to Computing.* Prentice-Hall, Englewood Cliffs, N. J., 1966, 212 pp.

   Text on fundamentals of algorithms, basic features of stored-program computers, and techniques involved in implementing algorithms on computers. Presents a complete description of FORTRAN IV with examples of numerical methods, nonnumerical applications, and simulations. Numerous exercises.

7. MARCOVITZ, A. B. AND SCHWEPPE, E. J. *An Introduction to Algorithmic Methods Using the MAD Language.* Macmillan, New York, 1966, 433 pp. CR-6781-11,199.

   Emphasizes algorithms and their expression as programs, characteristics of computers and computer systems, formal definition of computer languages, and accuracy and efficiency of programs. Numerous examples and exercises.

8. PERLIS, A. J. Programming for digital computers. *Comm. ACM* 7, 4 (Apr. 1964), 210-211.

   Description of course developed by Perlis at Carnegie Institute of Technology which has strongly influenced the course proposed here.

9. RICE, J. K. AND RICE, J. R. *Introduction to Computer Science: Problems, Algorithms, Languages and Information*, Preliminary edition. Holt, Rinehart and Winston, New York, 1967, 452 pp.

   Presentation revolves around the theme of "problem solving," emphasizing algorithms, languages, information representations, and machines necessary to solve problems. Problem solution methods classified, and many sample problems included. The nature of errors and uncertainty is considered. Detailed appendix on FORTRAN IV by E. Desautels.

10. School Mathematics Study Group. *Algorithms, Computation and Mathematics*, rev. ed. Stanford University, Stanford, Calif., 1966. *Student Text*, 453 pp., *Teacher's Commentary*, 301 pp.; *Algol Supplement: Student Text*, 133 pp., *Teacher's Commentary*, 109 pp.; *Fortran Supplement: Student Text*, 132 pp., *Teacher's Commentary*, 102 pp. Available from A. C. Vroman, Inc., 367 South Pasadena, Pasadena, Calif. *A MAD Language Supplement* by E. I. Organick is available from Ulrich's Book Store, 549 E. University Avenue, Ann Arbor, Mich.

    Although developed for high school students and teachers, this work contains much material appropriate for this course. Develops an understanding of the relationship between mathematics, computing, and problem solving. Basic text uses English and flow charts to describe algorithms; supplements introduce the computer language and give these algorithms in ALGOL, FORTRAN, and MAD.

Course B2. Computers and Programming (2-2-3)

APPROACH

This course is designed to introduce the student to basic computer organization, machine language programming, and the use of assembly language programming systems. A particular computer, machine language and programming system should be used extensively to illustrate the concepts being taught and to give the student actual experience in programming. However, it is important that the course not degenerate into mere training in how to program one machine. Alternative machine languages, machine organization, and programming systems should be discussed and compared. Emphasis should be placed on the overall structure of the machines and programming techniques considered. A "descriptive" presentation of various computer features and organizations may be very effective; nevertheless, it is recommended that a precise language be introduced and used to describe computer organizations and instruction execution (as the Iverson notation has been used to describe the IBM System/360).

CONTENT

The following outline indicates a possible order in which the material for this course might be taught, but other arrangements might be equally suitable depending upon the choice of text, availability of computing facilities, and preferences of the instructor. Computer projects—although not specifically listed below—should be an essential part of the course content.

1. *Computer Structure and Machine Language.* Organization of computers in terms of input-output, storage, control, and processing units. Register and storage structures, instruction format and execution, principal instruction types, and machine language programming. Machine arithmetic, program control, input-output operations, and interrupts. Characteristics of input-output and storage devices. (10%)

2. *Addressing Techniques.* Absolute addressing, indexing, indirect addressing, relative addressing, and base addressing. Memory mapping functions, storage allocation, associative addressing, paging, and machine organization to facilitate modes of addressing. (5%)

3. *Digital Representation of Data.* Bits, fields, words, and other information structures. Radices and radix conversion, representation of integer, floating-point, and multiple-precision numbers in binary and decimal form, and round-off errors. Representation of strings, lists, symbol tables, arrays and other data structures. Data transmission, error detection and correction. Fixed versus variable word lengths. (10%)

4. *Symbolic Coding and Assembly Systems.* Mnemonic operation codes, labels, symbolic addresses and address expressions. Literals, extended machine operations, and pseudo operations. Error flags and messages, updating, and program documentation. Scanning of symbolic instructions and symbol table construction. Overall design and operation of assemblers. (10%)

5. *Selected Programming Techniques (chosen from among the following).* Techniques for sorting, searching, scanning, and converting data. String manipulation, text editing, and list processing. Stack management, arithmetic expression recognition, syntactic recognition, and other compilation techniques. (10%)

6. *Logic Design, Micro-programming, and Interpreters.* AND, OR, and NOT elements, design of a half-adder and an adder, storage and delay elements, and design of an arithmetic unit. Parallel versus serial arithmetic, encoding and decoding logic, and micro-programming. Interpreters, simulation, and emulation. Logical equivalence between hardware and software. (5%)

7. *Macros.* Definition, call, and expansion of macros. Nested and recursive macro calls and definitions. Parameter handling, conditional assembly, and assembly time computations. (10%)

8. *Program Segmentation and Linkage.* Subroutines, coroutines, and functions. Subprogram loading and linkage, common data linkage, transfer vectors, and parameters. Dynamic storage allocation,

overlays, re-entrant subprograms, and stacking techniques. Linkage using page and segment tables. (10%)

9. *Computer Systems Organization.* Characteristics and use of tapes, disks, drums, cores, data-cells, and other large-volume devices in storage hierarchies. Processing unit organization, input-output channels and devices, peripheral and satellite processors, multiple processor configurations, computer networks, and remote access terminals. (10%)

10. *Systems and Utility Programs.* Loaders, input-output systems, monitors, and accounting programs. Program libraries. Organization, documentation, dissemination, and maintenance of system programs. (10%)

11. *Recent Developments.* Selected topics in computer organization, technology, and programming systems. (5%)

12. *Examinations.* (5%)

ANNOTATED BIBLIOGRAPHY

Whereas many of the books on "computer programming" might seem to be appropriate texts or references for this course, only a few even begin to approach the subject as proposed for this course. Most books deal with specific machines, actual or hypothetical, but very few discuss computer organization from any general point of view or consider the techniques of symbolic programming by any method other than examples. A few of the many books which deal with specific machines have been included in this list, but no manufacturers' manuals have been listed even though they may be used effectively as supplemental material.

1. Brooks, F. P., Jr., and Iverson, K. E. *Automatic Data Processing.* Wiley, New York, 1963, 494 pp. CR-6673-9523.

   On computing fundamentals, machine language organization and programming using IBM 650 as the principal example.

2. Davis, G. B. *An Introduction to Electronic Computers.* McGraw-Hill, New York, 1965, 541 pp.

   Informally written text containing a general introduction to computing, rather complete coverage of Fortran and Cobol, and considerable material on machines and machine language programming.

3. Fischer, F. P., and Swindle, G. F. *Computer Programming Systems.* Holt, Rinehart and Winston, New York, 1964, 643 pp. CR-6455-6299.

   Part I is concerned with machine oriented programming and programming systems using IBM 1401 as the illustrative computer.

4. Flores, I. *Computer Programming.* Prentice-Hall, Englewood Cliffs, N. J., 1966, 386 pp. CR-6674-10,060.

   Covers machine language and software techniques using the Flores Assembly Program (FLAP) for illustrative purposes.

5. Hassitt, A. *Computer Programming and Computer Systems.* Academic Press, New York, 1967, 374 pp. CR-6784-12,355.

   Discusses various features of computer organization and programming languages using examples from a number of machines including IBM 1401, 1620, 7090 and System/360, and CDC 1604 and 3600.

6. Iverson, K. E. *A Programming Language.* Wiley, New York, 1962, 286 pp. CR-6671-9004.

   Introduces a language used extensively for description of computers as well as for description of computer programs. Contains material on machine organization, sorting and data structures.

7. Stark, P. A. *Digital Computer Programming.* Macmillan, New York, 1967, 525 pp.

   Presents machine language and symbolic programming for a 24-bit computer.

8. Stein, M. L., and Munro, W. D. *Computer Programming: A Mixed Language Approach.* Academic Press, New York, 1964, 459 pp. CR-6455-6140.

   A text on computer organization and assembly language programming using CDC 1604 as the basic computer.

9. Wegner, P. *Programming Languages, Information Structures and Machine Organization.* McGraw-Hill, New York, 1968, about 410 pp.

   Covers machine languages, multiprogramming, assembler construction and procedure-oriented languages. Programming languages are treated as information structures.

Course B3. Introduction to Discrete Structures (3-0-3)

APPROACH

The theoretical material should be introduced in a mathematically precise manner with all concepts and results being amply motivated and being illustrated with examples from computer science. The student should be given extensive homework assignments of both a theoretical and a programming nature which further the understanding of the applications of the concepts in computer science.

CONTENT

Since the material listed below is more than can normally be offered in a one-semester three-credit course on this level, care must be taken to select those topics which will support the more advanced courses as they are developed at each particular school. The description in each of the four sections is divided into two parts labeled (a) Theory and (b) Applications, but in practice the material in both parts would be intermixed.

1. *Basic Set Algebra.*

a. Theory: Sets and basic set algebra. Direct products. Mappings, their domains and ranges, and inverse mappings. Finite and denumerable sets. Relations including order relations. Set inclusion as partial ordering. Equivalence relations, equivalence classes, partition of sets, congruences. The preservation of relations under mappings. Finite sets and their subsets. Permutations, combinations, and related combinatorial concepts.

b. Applications: Examples of sets. The Peano axioms for the set of integers. Congruences and ordering relations over the integers. Relations over the integers defined by arithmetic operations. The set of all subsets of an n-element set and the set of all n-digit binary numbers. The set of all strings over a finite alphabet. Languages over an alphabet as subsets of the set of all strings over the alphabet. Algorithms for listing combinations, compositions, or partitions. Algorithms for ranking combinations.

2. *Basic Algebraic Structures.*

a. Theory: Operations on a set. Algebraic structures as sets with particular functions and relations defined on it. Groups, subgroups, cyclic groups, and other examples of groups. The concepts of homomorphism and isomorphism on a set with operations. Semigroups and semigroups of transformations. Definition and general discussion of examples of structures with several operations, e.g. fields and possibly lattices.

b. Applications: Computer use for working group theoretic problems, e.g. with permutation groups as they occur as input transformation in switching networks. The semigroup of all words over a fixed finite alphabet under the operation of concatenation. The letters of the alphabet as generators. Pair algebra.

3. *Boolean Algebra and Propositional Logic.*

a. Theory: The axioms of set algebra. Axiomatic definition of Boolean algebras as algebraic structures with two operations. Duality. Basic facts about Boolean functions. Propositions and propositional functions. Logical connectives. Truth values and truth tables. The algebra of propositional functions. The Boolean algebra of truth values. Conjunctive and disjunctive normal forms.

b. Applications: Boolean algebra and switching circuits. Basic computer components. Decision tables.

4. *Graph Theory.*

a. Theory: Directed and undirected graphs. Subgraphs, chains, circuits, paths, cycles, connectivity, trees. Graphs and their relation to partial orderings. Graph isomorphisms. Cyclomatic and chromatic numbers. The adjacency and the incidence matrices. Minimal paths. Matchings of bipartite graphs. Elements of transport networks.

b. Applications: Flow charts and state transition graphs. Connectivity in flow charts. Syntactic structure of arithmetic expressions as trees. Graph theoretic examples in coding theory. Algorithms for determining cycles and minimal paths. Basic elements of list structures. Accessing problems. Graphs of a game. Matching algorithms and some related applications.

ANNOTATED BIBLIOGRAPHY

1. Beckenbach, E. F. (Ed.) *Applied Combinatorial Mathematics.* Wiley, New York, 1964, 608 pp.

   A collection of articles on a broad spectrum of topics. Not directly suitable as a text, but an excellent source of ideas and an important reference.

2. Berge, C. *Theory of Graphs and Its Applications.* Wiley, New York, 1962, 244 pp.

   A good presentation of directed and undirected graph theory, with some attention to algorithms. The work suffers from many misprints and errors which have been carried over into the English translation. A general reference text for this course.

3. Birkhoff, G., and Bartee, T. *Modern Applied Algebra*, Preliminary edition, *Parts I and II.* McGraw-Hill, New York, 1967.

   Preliminary edition available only in limited quantities, but the full text expected by the fall of 1968. Appears to be very close in spirit to the material proposed for this course, but the content is more algebraically oriented and includes little on graphs.

4. Busacker, R., and Saaty, T. *Finite Graphs and Networks: An Introduction with Applications.* McGraw-Hill, New York, 1965, 294 pp.

   A good work on graph theory with a very nice collection of applications. Useful as source and reference for the graph theory part of this course.

5. Grossman, I., and Magnus, W. *Groups and Their Graphs.* Random House, New York, 1965, 195 pp. CR-6564-8003.

   An elementary but very well written discourse on basic connections between group and graph theory.

6. Harary, F., Norman, R. Z., and Cartwright, D. *Structural Models: An Introduction to the Theory of Directed Graphs.* Wiley, New York, 1965, 415 pp. CR-6566-8421.

   Excellent on directed graphs and probably the best source book on that field. Should be an important reference for the corresponding portion of this course.

7. Hohn, F. *Applied Boolean Algebra*, 2nd ed. Macmillan, New York, 1966, 273 pp.

   Very good introduction to basic facts of Boolean algebra and especially its applications in electrical engineering. Important reference for the corresponding portion of this course.

8. Kemeny, J., Mirkil, H., Snell, J., and Thompson, G. *Finite Mathematical Structures.* Prentice-Hall, Englewood Cliffs, N. J., 1959, 487 pp.

   A text for physical science and engineering students who have completed the calculus. First two chapters on compound statements, sets, and functions should be particularly useful.

9. Kemeny, J., Snell, J., and Thompson, G. *Introduction to Finite Mathematics*, 2nd ed. Prentice-Hall, Englewood Cliffs, N. J., 1966, 352 pp.

   Freshman-sophomore level text designed primarily for students in biological and social sciences. Follows CUPM recommendations for the mathematical education of such students. First three chapters on compound statements, sets and subsets, partitions, and counting cover similar material as proposed for this course.

10. Korfhage, R. *Logic and Algorithms: With Applications to the Computer and Information Sciences.* Wiley, New York, 1966, 194 pp. CR-6782-11,339.

    A fine new text introducing those basic topics from mathematical logic important in computer science—for instance Boolean algebra, Turing machines, and Markov algorithms. Written in the spirit which should pervade this course.

11. Lederman, W. *Introduction to the Theory of Finite Groups.* Interscience, New York, 1953, 160 pp.

    A very readable introduction to finite groups. Particularly interesting to this course is the chapter on permutation groups.

12. MacLane, S., and Birkhoff, G. *Algebra.* Macmillan, New York, 1967, 598 pp.

    A substantially revised and updated version of *A Survey of Modern Algebra*, which has been a classic text on modern algebra. Should be one of the main references for the algebraic parts of this course.

13. Ore, O. *Graphs and Their Uses.* Random House, New York, 1963, 131 pp.

    An introduction to the elementary concepts of graph theory. Very pleasant to read.

14. Riordan, J. *An Introduction to Combinatorial Analysis.* Wiley, New York, 1958, 244 pp.

    One of the best source books on enumerative combinatorial analysis. However, it is too advanced for use as a text in a course of this type.

15. Ryser, H. *Combinatorial Mathematics.* Wiley, New York, 1963, 154 pp. CR-6562-7371.

    An excellent introduction to such topics as (0,1) matrices, Latin-squares, and block-design, but containing almost no graph theory.

16. Whitesitt, J. E. *Boolean Algebra and Its Applications.* Addison-Wesley, Reading, Mass., 1961, 182 pp.

    An introductory text designed for readers with a limited mathematical background.

Course B4. Numerical Calculus (2-2-3)

APPROACH

In this course the emphasis is placed upon building algorithms for the solution of numerical problems, the sensitivity of these algorithms to numerical errors, and the efficiency of these algorithms. In the laboratory portion of the course the student is to complete a substantial number of computational projects using a suitable procedure-oriented language.

CONTENT

1. *Basic Concepts of Numerical Error.* Significant digit arithmetic rounding procedures. Classification of error, evaluation of expressions and functions.

2. *Interpolation and Quadrature.* Polynomial interpolation, elements of difference calculus, Newton and Lagrange formulas, Aitken's interpolation method, quadrature formulas, Romberg integration, numerical differentiation, and the inherent error problems.

3. *Solution of Nonlinear Equations.* Bisection method, successive approximations including simple convergence proofs, linearization and Newton's method, method of false-position. Applications to polynomial equations. Generalization to iterative methods for systems of equations.

4. *Linear Systems of Equations.* Solution of linear systems and determinant evaluation by elimination procedures. Roundoff errors and ill-conditioning. Iterative methods.

5. *Numerical Solution of Ordinary Differential Equations.* Euler's method, modified Euler's method, simplified Runge-Kutta.

ANNOTATED BIBLIOGRAPHY

Listed below are some of the books which might be used as texts and/or references for this course. Most of the books cover the following topics: solution of polynomial and other nonlinear equations;

interpolation, numerical quadrature, and numerical differentiation; ordinary differential equations; and linear algebra. Significant deviations from these topics are indicated by the annotation.

1. Conte, S. D. *Elementary Numerical Analysis: An Algorithmic Approach.* McGraw-Hill, New York, 1965, 278 pp.

   Designed as a text for a one-semester, three-hour course for engineering and science undergraduate students. Machine-oriented treatment with many illustrative examples including flow charts and Fortran programs. Except for the chapter on differential equations, a knowledge of basic calculus and of programming in a procedure-oriented language is sufficient background. Numerous exercises.

2. Jennings, W. *First Course in Numerical Methods.* Macmillan, New York, 1964, 233 pp. CR-6671-9036.

   Designed as a text for a one-semester course for advanced undergraduate students in science and engineering. Brief treatment of the standard topics. Presupposes calculus, differential equations, some experience with the computer, and, for later chapters, matrices. Some exercises.

3. Macon, N. *Numerical Analysis.* Wiley, New York, 1963, 161 pp.

   Designed as a text for a one-semester first course in numerical analysis. Emphasis is more on the mathematical aspects rather than the computational aspects although there is an introductory chapter on the elements of computing, flow charting, and Fortran programming. For the early chapters calculus provides sufficient background. For later chapters an elementary knowledge of matrix theory, differential equations, and advanced calculus is recommended. Examples and exercises.

4. McCormick, J. M., and Salvadori, M. G. *Numerical Methods in FORTRAN.* Prentice-Hall, Englewood Cliffs, N.J., 1964, 324 pp. CR-6676-10,883.

   Designed as a text either for an elementary course in numerical analysis at the junior-senior level or for a course in programming. First part presents the methods without reference to programming techniques. There are 320 examples and problems. The last part contains 53 completely worked illustrative Fortran programs. Presupposes beginning analysis.

5. McCracken, D., and Dorn, W. S. *Numerical Methods and FORTRAN Programming.* Wiley, New York, 1964, 457 pp. CR-6562-7107.

   Designed as a text for a four semester-hour course in science or engineering at the sophomore-senior level. Emphasis on practical methods—for example, the treatment of simultaneous linear algebraic equations does not make use of matrices. Chapters on various aspects of Fortran are interspersed with chapters on numerical methods. Includes a brief chapter on partial differential equations. Presupposes beginning analysis. Examples and exercises.

6. Milne, W. E. *Numerical Calculus.* Princeton University Press, Princeton, N. J., 1949, 393 pp.

   Written in 1949 in the early days of computing, this is a very useful reference even though the treatment is oriented toward manual computation and though some of the methods have been superseded. Presupposes a knowledge of calculus and differential equations. Examples and exercises.

7. Nielsen, K. L. *Methods in Numerical Analysis,* 2nd ed. Macmillan, New York, 1956 and 1964, 382 pp. CR-6455-6333.

   Designed as a textbook for a practical course for engineers. Primary emphasis on the use of desk calculators and tables. Presupposes calculus. Examples and exercises.

8. Pennington, R. H. *Introductory Computer Methods and Numerical Analysis.* Macmillan, New York, 1965, 452 pp. CR-6565-8060.

   Designed as a text for a one-year elementary course for scientists and engineers to be taken immediately after integral calculus. The first part treats digital computers and programming. Numerical methods are then discussed from a computer viewpoint with the aid of flow diagrams. Little knowledge of computing is assumed. For some of the topics a knowledge of matrices and ordinary differential equations would be helpful. Many examples and exercises.

9. Singer, J. *Elements of Numerical Analysis.* Academic Press, New York, 1964, 395 pp. CR-6561-6959.

   Designed as a text for junior undergraduate students in mathematics. Treatment geared more to manual computation than to the use of computers. Presupposes beginning analysis and, for some parts, differential equations and advanced calculus. Examples and exercises.

10. Stiefel, E. L. *An Introduction to Numerical Mathematics,* transl. by W. C. and C. J. Rheinboldt. Academic Press, New York, 1963, 286 pp. CR-6455-6335.

    Appropriate for a junior-senior level course in mathematics, science, and engineering. Emphasis is on the algorithmic approach, although there are only a few flow charts and specific references to programs. A wide variety of topics and methods is treated. Basic calculus is required for the early chapters, but for later chapters familiarity with ordinary differential equations is desirable. Examples are given. There is a separate problem supplement with 36 exercises.

Course I1. Data Structures (3-0-3)

APPROACH

This course is intended to present the data structures which may be used in computer storage to represent the information involved in solving problems. However, emphasis should be placed on treating these data structures independently of the applications in which they are embedded. Each data structure should be motivated carefully in terms of the operations which may conveniently be performed, and illustrated with examples in which the structure is useful. The identification of the natural relations between entities involved in problems and alternate representations of information should be stressed. Computer storage structures should also be described and classified according to their characteristics, and the interaction between data structures and storage structures should be studied.

The student should be required to apply the techniques presented to problems which illustrate a wide variety of data structures. Solutions to a number of these problems should be programmed and run on a computer.

CONTENT

More material is listed here than can normally be covered in a one-semester course. The instructor should carefully select material which gives the student a broad introduction to this subject, but which fits together pedagogically. It may be desirable to develop an advanced course to cover some of these topics more completely.

1. *Basic Concepts of Data.* Representation of information as data inside and outside the computer. Bits, bytes, fields and data items. Records, nodes and data elements. Data files and tables. Names, values, environments, and binding times of data. Use of pointer or linkage variables to represent data structure. Identifying entities about which data is to be maintained, and selecting data nodes and structures which are to be used in problem solution. Storage media, storage structures, encoding of data and transformations from one medium and/or code to another. Alternative representations of information and data. Packing, unpacking, and compression of data. Data formats, data description languages, and specification of data transformations.

2. *Linear Lists and Strings.* Stacks, last-in-first-out, first-in-first-out, double-ended, and other linear lists. Sequential versus linked storage allocation. Single versus double linkage. Circular lists. Character strings of variable length. Word packing, part-word addressing, and pointer manipulation. Insertion, deletion and accessing of list elements.

3. *Arrays and Orthogonal Lists.* Storage of rectangular arrays in one-dimensional media. Storage mapping functions, direct and in-

direct address computation, space requirements, set-up time, accessing time, and dynamic relocation time. Storage and accessing triangular arrays, tetrahedral arrays, and sparse matrices.

4. *Tree Structures.* Trees, subtrees, ordered trees, free trees, oriented trees and binary trees. Representation of trees using binary trees, sequential techniques, or threaded lists. Insertion, deletion, and accessing elements of trees. Relative referencing, finding successors and predecessors, and walking through trees. Examples of tree structures such as algebraic formulas, arrays, and other hierarchic data structures (PL/I and COBOL).

5. *Storage Systems and Structures.* Behavioral properties of unit record (card), random access (core), linear (tape), and intermediate (disk, drum, etc.) storage media and devices including cost, size, speed, reusability, inherent record and file structure, and deficiencies and interrelation of these properties. Influence of machine structure—in particular addressing—on data structuring. Hierarchies of storage, virtual memory, segmentation, paging, and bucketing. Influence of data structures and data manipulation on storage systems. Associative structures, both hardware and software.

6. *Storage Allocation and Collection.* Static versus dynamic allocation. Sequential versus linked allocation. Last-in-first-out data versus data of unrelated life times. Uniform block size and available space lists. Variable block size and stratified available space lists. Explicit release of available storage. Coalescing adjacent free space and compacting occupied space or data. Accessing disciplines for movable data, unmovable anchors, and updating of pointers. Reference counts and list borrowing. Garbage collection by surveying active data.

7. *Multilinked Structures.* Use of different types of data nodes or elements. Use of different types of linkage to sequence, adjoin, or associate data elements and to build hierarchies of data structures. Sublists, list names, list heads, and attribute lists. Multidimensional linked lists and mixed list structures. Accessing, insertion, deletion and updating. Relative referencing, finding successors and predecessors, and walking through structures. Representation of graphs and networks. Structures used for string manipulation and list processing languages.

8. *Sorting (Ordering) Techniques.* Radix sorting, radix exchange sorting, merge sorting, bubble sorting, address table sorting, topological sorting and other sorting methods. Comparative efficiency of sorting techniques. Effect of data structures and storage structures on sorting techniques.

9. *Symbol Tables and Searching.* Linear, stack, tree and scatter structured tables, and table lookup techniques. Hash code algorithms. Use of index lists and associative techniques. Comparison of search strategies in terms of speed and cost. Batching and ordering of requests to remote storage to minimize number of accesses. TRIE memory as an example of structure organized for searching.

10. *Data Structures in Programming Languages.* Compile-time and run-time data structures needed to implement source language data structures of programming languages. Linkage between partially executed procedures, data structures for coroutines, scheduled procedures, and other control structures, and storage management of data structures in procedure-oriented languages. Examples of higher level languages which include list processing and other data structuring features.

11. *Formal Specification of Data Structures.* Specification of syntax for classes of data structures. Predicate selectors and constructors for data manipulation, data definition facilities, programs as data structures, computers as data structures and transformations, formal specification of semantics, and formal systems viewed as data structures.

12. *Generalized Data Management Systems.* Structures of generalized data management systems: directory maintenance, user languages (query), data description maintenance, and job management. Embedding data structures in generalized data management systems. Examples of generalized data management systems and comparison of system features.

ANNOTATED BIBLIOGRAPHY

Although a great deal of material is available in this area, very little of it is appropriate for classroom use.

1. Association for Computing Machinery. ACM sort symposium, Nov. 29–30, 1962, Princeton, N. J. *Comm. ACM 6*, 5 (May 1963), 194–272.
   Seventeen papers on various aspects of sorting.

2. Association for Computing Machinery. Papers presented at the ACM Storage Allocation Symposium, June 23–24, 1961, Princeton, N. J. *Comm. ACM 4*, 10 (Oct. 1961), 416–464.
   Eleven papers on various techniques of storage allocation.

3. Association for Computing Machinery. Proceedings of the ACM Symposium on Symbolic and Algebraic Manipulation, Washington, D. C., Mar. 29–31, 1966. *Comm. ACM 9*, 8 (Aug. 1966), 547–643.
   Eleven papers some of which discuss applications of data structuring techniques. One paper by Knowlton describes the list language L^6.

4. CLIMENSON, W. D. File organization and search techniques. In C. A. Cuadra (Ed.), *Annual Review of Information Science and Technology, Vol. 1*, (Amer. Doc. Inst., Ann. Rev. ser.), Interscience, New York, 1966, pp. 107–135. CR-6783-11,900.
   Surveys file organizations and data structures with particular emphasis on developments during 1965. Provides framework for some of the material covered by this course. An extensive bibliography.

5. COHEN, J. A use of fast and slow memories in list-processing languages. *Comm. ACM 10*, 2 (Feb. 1967), 82–86.
   Describes a paging scheme which keeps the "most often called pages in the fast memory" and involves a slow down of 3 to 10 as compared with in-core operations.

6. Control Data Corporation. *3600/3800 INFOL Reference Manual.* Publication No. 60170300, CDC, Palo Alto, Calif., July 1966.
   Describes the *INF*ormation *O*riented *L*anguage which is designed for information storage and retrieval applications.

7. DAHL, O.-J., AND NYGAARD, K. SIMULA—an ALGOL-based simulation language. *Comm. ACM 9*, 9 (Sept. 1966), 671–678.
   Contains interesting data and control structures.

8. D'IMPERIO, M. Data structures and their representation in storage. In M. Halpern (Ed.), *Annual Review in Automatic Programming, Vol. 5*, Pergamon Press, New York, spring 1968.
   Defines certain basic concepts involved in the representation of data and processes to be performed on data. Analyzes a problem and describes nine different solutions involving different data structures. Discusses ten list processing languages and gives examples of their data and storage structures.

9. FITZWATER, D. R. A storage allocation and reference structure. *Comm. ACM 7*, 9 (Sept. 1964), 542–545. CR-6561-6933.
   Describes a method of structuring and referencing dynamic structures in AUTOCODER for the IBM 7070/72/74.

10. General Electric Company. *Integrated Data Store—A New Concept in Data Management.* Application Manual AS-CPB-483A, Revision of 7-67, GE Computer Division, Phoenix, Ariz., 1967.
    Describes a sophisticated data management system which uses paging and chaining to develop complex data structures.

11. GRAY, J. C. Compound data structures for computer-aided design: a survey. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 355–365.
    Considers requirements of a data structure software package and surveys a number of such packages.

12. HELLERMAN, H. Addressing multidimensional arrays. *Comm. ACM 5*, 4 (Apr. 1962), 205–207. CR-6235-2619.
    Surveys direct and indirect methods for accessing arrays.

13. IVERSON, K. E. *A Programming Language.* Wiley, New York, 1962, 286 pp. CR-6671-9004.

Contains considerable material on data structures, graphs, trees, and sorting, as well as a language for describing these.

14. Klein, M. M. Scheduling project networks. *Comm. ACM 10*, 4 (Apr. 1967), 225-234. CR-6784-12,275.

Discusses project networking and describes the C-E-I-R critical path algorithm.

15. Knuth, D. E. *The Art of Computer Programming, Vol. 1, Fundamental Algorithms.* Addison-Wesley, Reading, Mass., 1968, 634 pp.

Chap. 2 on "Information Structures" contains the first comprehensive classification of data structures to be published. Each structure considered is carefully motivated and generously illustrated. Includes a brief history of data structuring and an annotated bibliography.

16. Landin, P. J. The mechanical evaluation of expressions. *Comput. J. 6*, 4 (Jan. 1964), 308-320. CR-6456-6677.

Presents a mathematical language based on Church's λ-notation and uses it to describe computational structures such as expressions and lists.

17. Lawson, H. W., Jr. PL/I list processing. *Comm. ACM 10*, 6 (June 1967), 358-367.

Discusses the list processing facilities in PL/I.

18. Madnick, S. E. String processing techniques. *Comm. ACM 10*, 7 (July 1967), 420-424.

Presents and evaluates six techniques for string data storage structures. One of these techniques is used for an implementation of Snobol on an IBM System/360.

19. Marron, B. A., and de Maine, P. A. D. Automatic data compression. *Comm. ACM 10*, 11 (Nov. 1967), 711-715.

Describes a three-part compressor which can be used on "any" body of information to reduce slow external storage requirements and to increase the rate of information transmission through a computer.

20. Mealy, G. H. Another look at data. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 525-534.

Sketches a theory of data based on relations. Includes some rather precise definitions of concepts such as data structure, list processing, and representation.

21. Minker, J., and Sable, J. File organization and data management. In C. A. Cuadra (Ed.), *Annual Review of Information Science and Technology, Vol. 2*, (Amer. Doc. Inst., Ann. Rev. ser.). Interscience, New York, 1967, pp. 123-160.

Surveys file organizations and generalized data management systems developed during 1966. Describes linkage types, data structures, storage structures, and how data structures have been mapped into storage structures. Extensive bibliography.

22. Morris, R. Scatter storage techniques. *Comm. ACM 11*, 1 (Jan. 1968), 38-44.

Surveys hashing schemes for symbol table algorithms.

23. Rosen, S. (Ed.) *Programming Languages and Systems.* McGraw-Hill, New York, 1967, 734 pp.

Part 4 of this collection contains papers on IPL-V, Comit, Slip, Snobol, Lisp and a comparison of list-processing computer languages.

24. Ross, D. T. The AED free storage package. *Comm. ACM 10*, 8 (Aug. 1967), 481-492.

Describes a storage allocation and management system for the mixed n-component elements ("beads") needed for "plex programming."

25. Salton, G. Data manipulation and programming problems in automatic information retrieval. *Comm. ACM 9*, 3 (Mar. 1966), 204-210. CR-6674-10,078.

Describes a variety of representations for tree structured data and examines their usefulness in retrieval applications.

26. Savitt, D. A., Love, H. H., Jr., and Troop, R. E. ASP: a new concept in language and machine organization. Proc. 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 87-102.

Describes the data bases used in the "Association-Storing Processor." These structures are complex in organization and may vary dynamically in both organization and content.

27. Schorr, H., and Waite, W. M. An efficient machine-independent procedure for garbage collection in various list structures. *Comm. ACM 10*, 8 (Aug. 1967), 501-506.

Reviews and compares past garbage collection methods and presents a new algorithm.

28. Standish, T. A. A data definition facility for programming languages. Ph.D. Thesis, Carnegie Institute of Technology, Pittsburgh, Pa., 1967.

Presents a descriptive notation for data structures which is embedded in a programming language.

29. Wegner, P. (Ed.) *Introduction to Systems Programming.* Academic Press, New York, 1965, 316 pp. CR-6455-6300.

Contains a collection of papers of which the following are of special interest for this course: Iliffe, pp. 256-275; Jenkins, pp. 283-293; and Burge, pp. 294-312.

30. Wegner, P. *Programming Languages, Information Structures, and Machine Organization.* McGraw-Hill, New York, 1968, about 410 pp.

Introduces information structures and uses them in describing computer organization and programming languages.

Course I2. Programming Languages (3-0-3)

APPROACH

This course is intended to survey the significant features of existing programming languages with particular emphasis on the underlying concepts abstracted from these languages. The relationship between source programs and their run-time representation during evaluation will be considered, but the actual writing of compilers is to be taught in Course I5.

CONTENT

There are four basic parts of this course: the structure of simple statements; the structure of algorithmic languages; list processing and string manipulation languages; and topics in programming languages.

Part A. Structure of Simple Statements. (10 lectures)

1. Informal syntax and semantics of arithmetic expressions and statements, translation between infix, prefix, and postfix notation, and the use of pushdown stores for translation and execution of arithmetic expressions and statements. Precedence hierarchy of arithmetic operations, relational operators, and Boolean operators. Backus normal form representation of syntax of arithmetic statements and the semantics of arithmetic statements. (6 lectures)

2. Precedence relations, precedence grammars, and syntactic analysis of precedence grammars. Application to arithmetic expressions, code generation, error diagnostics and error correction for syntactic arithmetic expression compilation. (4 lectures)

Part B. Structure of Algorithmic Languages. (20 lectures)

3. Review of program constituents, branching statements and loops. (2 lectures)

4. Grouping of statements, declarations, "types" of program constituents, nomenclature, scopes, local and nonlocal quantities, independent blocks (Fortran), and nested blocks (Algol). (2 lectures)

5. Function and statement type procedures, formal parameters and actual parameters, and call by value, name and reference. Binding time of program constituents, recursive procedures, and side effects during execution of procedures. (3 lectures)

6. Storage allocation for independent blocks (Fortran) and storage allocation for nested blocks and procedures using a run-time

pushdown store. Overall structure of an Algol-style compiler. (3 lectures)

7. Coroutines, tasks, interrupt specification, and classification of control structures in procedure-oriented languages. (2 lectures)

8. Syntactic specification of procedures, blocks and statements. Formal semantics corresponding to syntactic specification. Survey of principal concepts of syntactic analysis. (5 lectures)

9. Generalized arrays. Data definition facilities, pointer-valued variables, and list creation and manipulation using pointer-valued variables. Templates and controlled storage allocation. Distinction between data specification by a data template and the creation of instances of a specified data structure. (3 lectures)

Part C. List Processing and String Manipulation Languages. (7 lectures)

10. List structures, basic operations on list structures, Lisp-like languages, machine-oriented list processing languages (IPL-V), embedding of list operations in algorithmic languages (Slip), dynamic storage allocation for list languages, and garbage collection. (5 lectures)

11. String structures, operations on strings, and functions which have strings as arguments and strings as their values (Snobol). (2 lectures)

Part D. Topics in Programming Languages. (8 lectures)

12. Additional features of programming languages, simulation languages, algebraic manipulation languages, and languages with parallel programming facilities. (2–6 lectures)

13. Formal description of languages and their processors. The work of Floyd, Wirth, and others. (2–6 lectures)

14. Other topics selected by the instructor.

ANNOTATED BIBLIOGRAPHY

1. American Standards Association X3.4.1 Working Group. Toward better documentation of programming languages. *Comm. ACM 6*, 3 (Mar. 1963), 76–92.

   A series of papers describing the documentation of significant current programming languages.

2. Association for Computing Machinery. Proceedings of the ACM programming languages and pragmatics conference, San Dimas, Calif., August 8–12, 1965. *Comm. ACM 9*, 3 (Mar. 1966), 137–232.

   Includes a number of papers applicable to this course.

3. Association for Computing Machinery. Proceedings of the ACM symposium on symbolic and algebraic manipulation, Washington, D. C., March 29–31, 1966. *Comm. ACM 9*, 8 (Aug. 1966), 547–643.

   A number of languages for symbolic and algebraic manipulation are described in this special issue.

4. Dahl, O.-J., and Nygaard, K. SIMULA—an ALGOL-based simulation language. *Comm. ACM 9*, 9 (Sept. 1966), 671–678.

   Describes a language encompassing Algol, but having many additional features including those needed for simulation.

5. Galler, B. A., and Perlis, A. J. A proposal for definitions in ALGOL. *Comm. ACM 10*, 4 (Apr. 1967), 204–219.

   Describes a generalization of Algol which allows new data types and operators to be declared.

6. Goodman, R. (Ed.) *Annual Review in Automatic Programming, Vols. 1, 2, 3, 4.* Pergamon Press, New York, 1960 to 1965. CR-6123-0811, CR-6235-2602, and CR-6564-7901.

   These volumes contain several papers which are applicable to this course.

7. Halstead, M. H. *Machine-Independent Computer Programming.* Spartan Books, New York, 1962.

   Contains both internal and external specifications of the Neliac programming language.

8. IEEE Computer Group. The special issue on computer languages. *IEEE Trans. EC-13*, 4 (Aug. 1964), 343–462.

   Contains articles on Algol, Fortran, Formac, Sol and other computer languages.

9. International Business Machines. PL/I Language Specification. Form C28-6571-4, IBM System/360 Operating System, IBM Corporation, White Plains, N. Y., 1967.

   A specification of the PL/I language.

10. International Standards Organization Technical Committee 97, Subcommittee 5. Survey of programming languages and processors. *Comm. ACM 6*, 3 (Mar. 1963), 93–99.

    An international survey of current and imminent programming languages.

11. Knuth, D. E. The remaining trouble spots in ALGOL 60. *Comm. ACM 10*, 10 (Oct. 1967), 611–618.

    This paper lists the ambiguities which remain in Algol 60 and which have been noticed since the publication of the Revised Algol 60 Report in 1963.

12. Markowitz, H. M., Karr, H. W., and Hausner, B. *SIMSCRIPT: A Simulation Programming Language.* Prentice-Hall, Englewood Cliffs, N. J., 1963, 138 pp.

    A description of the Simscript simulation language. There is a new Simscript 1.5 supplement now available which describes a generalization of the original language.

13. Mooers, C. N. TRAC, a procedure-describing language for the reactive typewriter. *Comm. ACM 9*, 3 (Mar. 1966), 215–219. CR-6674-10,079.

    Describes a language for the manipulation of text from an on-line typewriter.

14. Naur, P. (Ed.) Revised report on the algorithmic language, ALGOL 60. *Comm. ACM 6*, 1 (Jan. 1963), 1–17. CR-6016-0323.

    The Backus normal form notation was developed to help describe the syntax of Algol in the original version of this report (*Comm. ACM 3*, 5 (May 1960), 299–314).

15. Perlis, A. J. The synthesis of algorithmic systems—first annual A. M. Turing lecture. *J. ACM 14*, 1 (Jan. 1967), 1–9. CR-6782-11,512.

    A stimulating talk on the nature of programming languages and the considerations which should underlie their future development.

16. Rosen, S. (Ed.) *Programming Systems and Languages.* McGraw-Hill, New York, 1967, 734 pp.

    This collection of papers contains many of the important references for this course. In particular, Parts 1 and 2 of the collection are useful for Parts A and B of the course and Part 4 of the collection is useful for Part C of the course.

17. Shaw, C. J. A comparative evaluation of JOVIAL and FORTRAN IV. *Automatic Programming Inf., No. 22.* Technical College, Brighton, England, Aug. 1964, 15 pp. CR-6562-7265.

    A descriptive point-by-point comparison of these two languages. Concerned mainly with the features of the languages rather than their processors.

18. Shaw, C. J. A programmer's look at JOVIAL, in an ALGOL perspective. *Datamation 7*, 10 (Oct. 1961), 46–50. CR-6233-1933.

    An interesting article showing how Algol and Jovial evolved from Algol 58 and how they differ.

19. USA Standards Institute. Standards X3.9-1966, FORTRAN and X3.10-1966, Basic FORTRAN. USASI, 10 East 40th Street, New York, N. Y. 10016, 1966.

    Standard definitions of essentially Fortran II and Fortran IV. These also appeared in almost final form in *Comm. ACM 7*, 10 (Oct. 1964), 591–625.

20. Wegner, P. *Programming Languages, Information Structures, and Machine Organization.* McGraw-Hill, New York, 1968, about 410 pp.

    Develops a unified approach to the study of programming languages emphasizing the treatment of such languages as infor-

mation structures. First two chapters devoted to machine organization, machine language, and assembly language, but much of Chap. 3 and essentially all of Chap. 4 devoted to the topics of this course.

21. Wirth, N. A generalization of ALGOL. *Comm. ACM 6*, 9 (Sept. 1963), 547–554. CR-6451-5030.

    Proposes a generalization of Algol which involves the elimination of "type" declarations and the replacement of procedure declarations by an assignment of a so-called "quotation."

22. Wirth, N., and Weber, H. EULER—a generalization of ALGOL and its formal definition, Parts I and II. *Comm. ACM 9*, 1 (Jan. 1966), 13–23, and 2 (Feb. 1966), 89–99.

    Develops a method for defining programming languages which introduces a rigorous relationship between structure and meaning. The structure of a language is defined by a phrase structure syntax and the meaning is defined in terms of the effects which the execution of a sequence of interpretation rules has upon a fixed set of variables called the "environment."

Course 13. Computer Organization (3-0-3) or (3-2-4)

APPROACH

This course is intended to introduce the student to the basic ideas of computer elements and logic design techniques and to the principles of computer systems organization. Emphasis should be placed on the various alternative possibilities which must be considered in arriving at a computer design; choices such as character or word organized data, serial or parallel data transmission, synchronous or asynchronous control should be compared and evaluated.

In addition to using block diagrams, it is recommended that a formal descriptive language for computer specification be introduced and used to provide a uniform method for the presentation of much of the material. The student should carry out a detailed computer design project and evaluate his design by simulation, if possible. A laboratory in which simple logic elements may be combined to perform digital functions is also desirable.

CONTENT

The following topics are to be covered, although not necessarily in the order listed.

1. *Basic Digital Circuits.* Switches, relays, transistors, diodes, magnetic cores, circuits of individual elements, and integrated circuits. (These topics may be spread throughout the course.) (5%)

2. *Boolean Algebra and Combinational Logic.* Boolean values, variables, operations, expressions and equations. Logic elements such as AND, OR, NOT, NAND and NOR. Correspondence between Boolean functions and combinations of logic elements. (5%)

3. *Data Representation and Transfer.* Flip-flops, registers, core storage, and other memory elements. Review of number representations, binary versus binary-coded decimal representation, and integer versus floating-point representation. Weighted and nonweighted codes, redundancy, and coding of character information. Coders and decoders. Clearing, gating and other transfer considerations. (10%)

4. *Digital Arithmetic.* Counters, comparators, parity checkers, and shift registers. Half and full adders. Serial versus parallel adders. Subtraction and signed magnitude versus complemented arithmetic. Multiplication and division algorithms. Integer versus floating-point arithmetic. Double precision arithmetic. Elementary speed-up techniques for arithmetic. (10%)

5. *Digital Storage and Accessing.* Structure of core memory, memory control, data buses, and address buses. Addressing and accessing methods including index registers, indirect addressing, base registers, and other techniques. Overlapping, interleaving, protection, dynamic relocation, and memory segmentation methods. Characteristics of drum, disk, tape, and other surface recording media and devices. Data flow in multimemory systems and hierarchies of storage. (10%)

6. *Control Functions.* Synchronous versus asynchronous control. Time pulse distributors, controlled delay techniques, and Gray code control sequencers. Instruction repertoire, decoding networks, and sequencing methods. Centralized, decentralized, and micro-programmed control units. Internal, external and trapping interrupts. Interrupt sensing, priority, selection, recognition, and processing. Input-output control. (10%)

7. *Input-Output Facilities.* Characteristics of input-output devices and their controllers. Relationship between input-output devices, main storage, auxiliary storage, buffers, data channels, and multiplexers. Serial versus parallel transmission. Low speed, high speed, and burst mode data flow. (10%)

8. *System Organization.* Overall organization of modules into a system. Interface between modules. Word-oriented versus character-oriented machines. Simplex and multiprocessor machines. Special purpose computers. Relationship between computer organization and software. (15%)

9. *Reliability.* Error detection and correction. Diagnostics and preventive maintenance. Start-up, power-down, and recovery procedures. (5%)

10. *Description and Simulation Techniques.* Definition of a formal computer description language which would be used in discussing most of the other topics listed for the course. Use of a computer simulator to design and test simple computers or computer modules. (10%)

11. *Selected Topics.* Multiple arithmetic units, instruction overlapping, and look-ahead techniques. Discussion of alternate organizations including highly parallel machines. (5%)

12. *Examinations.* (5%)

ANNOTATED BIBLIOGRAPHY

As indicated, several of the books listed below might possibly be used as texts for this course, but it probably would be good to supplement any of them with additional material. Only a few of the many references on computer description languages and programs for simulating computer designs are listed; no annotations are given for these.

General textbooks

1. Bartee, T. C. *Digital Computer Fundamentals.* McGraw-Hill, New York, 1960, 1966, 401 pp. CR-6676-10,647.

   Not advanced enough but a very useful supplement for circuits and equipment.

2. Bartee, T. C., Lebow, I. L., and Reed, I. S. *Theory and Design of Digital Systems.* McGraw-Hill, New York, 1962, 324 pp. CR-6344-4416.

   Very mathematical and somewhat out-of-date. An interesting reference.

3. Braun, E. L. *Digital Computer Design—Logic, Circuitry, and Synthesis.* Academic Press, New York, 1963, 606 pp. CR-6453-5484.

   Somewhat out-of-date for a text but useful as a reference.

4. Buchholz, W. *Planning a Computer System.* McGraw-Hill, New York, 1962, 336 pp. CR-6346-4786.

   Good reference on systems concepts but somewhat dated.

5. Burroughs Corporation. *Digital Computer Principles.* McGraw-Hill, New York, 1962, 507 pp.

   Restricted scope (engineering oriented) and dated, but could be used as a reference.

6. Chu, Y. *Digital Computer Design Fundamentals.* McGraw-Hill, New York, 1962, 481 pp. CR-6343-4198.

   Good reference which contains a wealth of material on logic design.

7. Flores, I. *Computer Logic.* Prentice-Hall, Englewood Cliffs, N. J., 1960, 458 pp. CR-6122-0641 and CR-6124-0936.

   Dated and unorthodox but possibly useful for supplementary reading.

8. Flores, I. *The Logic of Computer Arithmetic.* Prentice-Hall,

Englewood Cliffs, N. J., 1963, 493 pp. CR-6452-5458.

Very detailed, unorthodox treatment of computer arithmetic.

9. Gschwind, H. W. *Design of Digital Computers.* Springer-Verlag, New York, 1967.

A possible text.

10. Hellerman, H. *Digital Computer System Principles.* McGraw-Hill, New York, 1967, 424 pp.

A possible text. Uses Iverson notation throughout. Would have to be supplemented on circuits and equipment as well as novel organizations.

11. Maley, G. A., and Skiko, E. J. *Modern Digital Computers.* Prentice-Hall, Englewood Cliffs, N. J., 1964, 216 pp. CR-6561-7081.

A possible reference. Somewhat dated but contains a good description of the IBM 7090 and 7080 machines.

12. Murtha, J. C. Highly parallel information processing systems. In F. L. Alt (Ed.), *Advances in Computers, Vol. 7.* Academic Press, New York, 1966, pp. 1-116. CR-6782-11,678.

A useful reference on highly parallel systems.

13. Phister, M., Jr. *Logical Design of Digital Computers.* Wiley, New York, 1958, 408 pp.

Somewhat dated. Relies heavily on sequential circuit theory and concentrates on serial, clocked machines.

14. Richards, R. K. *Arithmetic Operations in Digital Computers.* D. Van Nostrand, Princeton, N. J., 1955, 397 pp.

Somewhat dated but still a good reference for arithmetic.

15. Richards, R. K. *Electronic Digital Systems.* Wiley, New York, 1966, 637 pp. CR-6676-10,649.

An interesting reference for reliability and design automation. Discusses telephone systems and data transmission.

References on computer description languages

16. Chu, Y. An ALGOL-like computer design language. *Comm. ACM 8*, 10 (Oct. 1965), 607-615. CR-6672-9315.

17. Falkhoff, A. D., Iverson, K. E., and Sussenguth, E. H. A formal description of System /360. *IBM Syst. J. 3*, 3 (1964), 198-263.

18. Gorman, D. F., and Anderson, J. P. A logic design translator. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22, Spartan Books, New York, pp. 251-261.

19. Iverson, K. E. *A Programming Language.* Wiley, New York, 1962, 286 pp. CR-6671-9004.

20. McClure, R. M. A programming language for simulating digital systems. *J. ACM 12*, 1 (Jan. 1965), 14-22. CR-6563-7634.

21. Parnas, D. L., and Darringer, J. A. SODAS and a methodology for system design. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 449-474.

22. Wilber, J. A. A language for describing digital computers. M.S. Thesis, Report No. 197, Dept. of Comput. Sci., U. of Illinois, Urbana, Ill., Feb. 15, 1966.

Course I4. Systems Programming (3-0-3)

APPROACH

This course is intended to bring the student to grips with the actual problems encountered in systems programming. To accomplish this it may be necessary to devote most of the course to the study of a single system chosen on the basis of availability of computers, systems programs, and documentation.

The course should begin with a thorough review of "batch" processing systems programming, emphasizing loading and subroutine linkage. The limitations of these systems should be used to motivate the more complex concepts and details of multiprogramming and multiprocessor systems. The theoretical concepts and practical techniques prescribed in Course I1 should be used to focus on the data bases, their design for the support of the functions of the key system components (hardware and software), and the effective interrelation of these components. Problem assignments should involve the design and implementation of systems program modules; the design of files, tables or lists for use by such modules; or the critical use and evaluation of existing system programs. Other problems might involve the attaching or accessing of procedure or data segments of different "ownership" that are resident in a single file system or the development of restricted accessing methods (i.e. privacy schemes) and other such techniques.

CONTENT

This description has been written with MULTICS in mind as the system chosen for central study, but the description can be modified to fit any reasonably comprehensive system. There is considerably more material listed here than can normally be covered in one semester, so that careful selection of topics should be made or the course should be extended to two semesters.

1. *Review of Batch Process Systems Programs.* Translation, loading, and execution. Loader languages. Communication between independent program units. Limitations imposed by binding at pre-execution times. Incremental linkage.

2. *Multiprogramming and Multiprocessor Systems.* General introduction to the structure of these systems, the techniques involved in their construction and some of the problems involved in their implementation.

3. *Addressing Techniques.* Review of indexing and indirect addressing. Relocation and base registers. Two-dimensional addressing (segmentation). Segmented processes. Concepts of virtual memory. Effective address computation. Modes of access control. Privileged forms of accessing. Paging. Physical register (address) computation, including use of associative memories.

4. *Process and Data Modules.* Concept of a process as a collection of procedure and data components (segments). Process data bases. Controlled sharing of segments among two or more processes. Intersegment linking and segment management. Interprocedure communication. Process stacks. Levels of isolation within a process (rings of protection).

5. *File System Organization and Management.* File data bases and their storage structures. Accessing, protection and maintenance of files. Storage and retrieval of segments and/or pages from files in secondary storage (segment and page control, directory control, and core management). Search strategies.

6. *Traffic Control.* State words. Running, ready, blocked, and inactive processes. Process switching. Priority control of waiting processes. Scheduling algorithms. Pseudo processes. System tables for process management.

7. *Explicit Input-Output References.* Auxiliary (secondary) memory references. Communication with peripheral devices. Management of input-output and other request queues. Effects of data rates on queue management.

8. *Public and Private Files.* On line and off line memory. Automatic shifting of data among devices in the storage hierarchy and "flushing" of online memory, i.e. multilevel storage management. File backup schemes and recovery from system failures.

9. *Other Topics.* Some of the following topics may be studied if time permits. They might also be covered in subsequent seminars.

a. System accounting for facilities employed by the user. Special hardware features for metering different uses. Accounting for system overhead. Factors which determine system overhead.

b. Characteristics of large systems. Overall discussion of large system management including effect of binding times for system and user process variables. Other selected topics on large systems such as the effect of new hardware components (e.g. mass memories) on the overall system design.

c. Foreground and background processes. Foreground-initiated background processes. Remote job control. Hierarchical job control. Broadcasting.

d. Microprogramming as an equivalent of various hardware and/or software component subprograms in a computing system.

e. Command languages. Commands of a multiprogramming system. Command language interpreters.

f. Provisions for dynamic updating of the operating system without shutdown.

g. Operating behavior, e.g. system startup, (graceful) degradation, and shutdown.

ANNOTATED BIBLIOGRAPHY

In addition to the following sources of information, there are many manuals available from manufacturers which describe specific systems programs for a wide range of computers.

1. CHOROFAS, D. N. *Programming Systems for Electronic Computers.* Butterworths, London, 1962, 188 pp. CR-6566-8553.

   Chapters 14, 15, and 16 contain a general discourse on the control and diagnostic functions of operating systems.

2. CLARK, W. A., MEALY, G. H., AND WITT, B. I. The functional structure of OS/360. *IBM Syst. J. 5*, 1 (1966), 3-51.

   A general description in three parts of the operating system for the IBM System/360. Part I (Mealy), Introductory Survey; Part II (Witt), Job and Task Management; and Part III (Clark), Data Management.

3. DESMONDE, W. H. *Real-Time Data Processing Systems. Introductory Concepts.* Prentice-Hall, Englewood Cliffs, N. J., 1964, 192 pp. CR-6562-7236.

   An elementary survey of the design and programming of real-time data processing systems based on three IBM systems: Sabre, Mercury, and Gemini.

4. ERDWINN, J. D. (Ch.) Executive control programs—Session 8. Proc. AFIPS 1967 Fall Joint Comput. Conf., Vol. 31, Thompson Book Co., Washington, D. C., pp. 201-254.

   Five papers on control programs for a variety of circumstances.

5. FISCHER, F. P., AND SWINDLE, G. F. *Computer Programming Systems.* Holt, Rinehart and Winston, New York, 1964, 643 pp. CR-6455-6299.

   Sets out to "discuss the entire field of computer programming systems," but in reality considers primarily the systems programs for the IBM 1401; "other computer systems are mentioned only where a particular characteristic of a programming system, found on that computer, warrants discussion." Only IBM computer systems and (with a very few exceptions) only IBM literature are referenced.

6. FLORES, I. *Computer Software.* Prentice-Hall, Englewood Cliffs, N. J., 1965, 464 pp. CR-6671-8995.

   Elementary and conversational text primarily concerned with assembly systems using FLAP (Flores Assembly Program) as its example. Some material on service programs, supervisors and loaders.

7. GLASER, E. (Ch.) A new remote accessed man-machine system—Session 6. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Pt. 1, Spartan Books, New York, pp. 185-241. (Reprints available from the General Electric Company.)

   Six papers on the MULTICS system.

8. HEISTAND, R. E. An executive system implemented as a finite-state automaton. *Comm. ACM 7*, 11 (Nov. 1964), 669-677. CR-6562-7282.

   Describes the executive system for the 473L command and control system. The system was considered as a finite automaton and the author claims this approach forced a modularity on the resulting program.

9. LEONARD, G. F., AND GOODROE, J. R. An environment for an operating system. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. E2.3-1 to E2.3-11. CR-6561-6546.

   An approach to computer utilization involving the extension of the operations of a computer with software so as to provide a proper environment for an operating system.

10. MARTIN, J. *Design of Real-Time Computer Systems.* Prentice-Hall, Englewood Cliffs, N. J., 1967, 629 pp.

    A general text covering many aspects of real-time data processing systems including design, applications, management, and operation.

11. MARTIN, J. *Programming Real-Time Computer Systems.* Prentice-Hall, Englewood Cliffs, N. J., 1965, 386 pp.

    Based on some of the early systems such as Sage, Project Mercury, Sabre, and Panamac. A general coverage designed for managers, systems analysts, programmers, salesmen, students.

12. MILLER, A. E. (Ch.) Analysis of time-shared computer system performance—Session 5. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 85-109.

    Three papers on measurement of time-shared system performance.

13. M.I.T. Computation Center. *Compatible Time-Sharing System: A Programmer's Guide*, 2nd ed. M.I.T. Press, Cambridge, Mass., 1965.

    A handbook on the use of CTSS which contains valuable information and guidelines on the implementation of such systems.

14. Project MAC. *MULTICS System Programmer's Manual.* Project MAC, M.I.T., Cambridge, Mass., 1967, (limited distribution).

    A description of and guide to systems programming for MULTICS.

15. ROSEN, S. (Ed.) *Programming Systems and Languages.* McGraw-Hill, New York, 1967, 734 pp.

    Collection of important papers in the area of which Part 5 (Operating Systems) is particularly relevant to this course.

16. ROSENBERG, A. M. (Ch.) Program structures for the multiprogramming environment—Session 6A. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 223-239.

    Two papers: one on program behavior under paging; the other on analytic design of look-ahead and program segmenting systems.

17. ROSENBERG, A. M. (Ch.) Time-sharing and on-line systems—Session 7. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 135-175.

    Three papers on various topics related to the subject.

18. SALTZER, J. H. Traffic control in a multiplexed computer system. M.I.T. Ph.D. Thesis, June 1966. (Also available as Project MAC publication MAC-TR-30.)

    On traffic control in the MULTICS system.

19. SMITH, J. W. (Ch.) Time-shared scheduling—Session 5A. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 139-177.

    Four papers on time-sharing which are more general than the session title indicates.

20. THOMPSON, R. N., AND WILKINSON, J. A. The D825 automatic operating and scheduling program. Proc. AFIPS 1963 Spring Joint Comput. Conf., Vol. 23, Spartan Books, New York, pp. 41-49. CR-6453-5699.

    A general description of an executive system program for handling a multiple computer system tied to an automatic input-output exchange containing a number of input-output control modules. Discusses many of the problems encountered in such systems and the general plan of attack in solving these problems.

21. WEGNER, P. (Ed.) *Introduction to System Programming.* Academic Press, New York, 1965, 316 pp. CR-6455-6300.

    Contains a collection of papers of which the following are of special interest for this course: Gill, pp. 214-226; Howarth, pp. 227-238; and Nash, pp. 239-249.

Course I5. Compiler Construction (3-0-3)

APPROACH

This course is to emphasize the techniques involved in the analysis of source language and the generation of efficient object code. Although some theoretical topics must be covered, the course should have the practical objective of teaching the student how compilers may be constructed. Programming assignments should consist of implementations of components of a compiler and possibly the design of a simple but complete compiler as a group project.

CONTENT

There is probably more material listed here than can reasonably be covered, so some selection will be necessary.

1. Review of assembly techniques, symbol table techniques, and macros. Review of syntactic analysis and other forms of program recognition. Review of compilation, loading, and execution with emphasis on the representation of programs in the loader language.

2. One-pass compilation techniques. Translation of arithmetic expressions from postfix form to machine language. Efficient use of registers and temporary storage.

3. Storage allocation for constants, simple variables, arrays, temporary storage. Function and statement procedures, independent block structure, nested block structure, and dynamic storage allocation.

4. Object code for subscripted variables, storage mapping functions, and dope vectors. Compilation of sequencing statements.

5. Detailed organization of a simple complete compiler. Symbol tables. Lexical scan on input (recognizer), syntax scan (analyzer), object code generators, operator and operand stacks, output subroutines, and error diagnostics.

6. Data types, transfer functions, mixed mode expressions and statements.

7. Subroutine and function compilation. Parameters called by address, by name and by value. Subroutines with side effects. Restrictions required for one pass execution. Object code for transmission of parameters. Object code for subroutine body.

8. Languages designed for writing compilers: TMG (McClure), COGENT (Reynolds), GARGOYLE (Garwick), META II (Schorre), and TGS-II (Cheatham).

9. Bootstrapping techniques. Discussion of a meta-compiler in its own language.

10. Optimization techniques. Frequency analysis of use of program structures to determine most important features for optimization.

11. Local optimization to take advantage of special instructions. Loading registers with constants, storing zeros, changing sign, adding to memory, multiplication or division by two, replacement of division with multiplication by a constant, squaring, raising to integer powers, and comparing to zero. Subscript optimization.

12. Expression optimization. Identities involving minus signs, common subexpression evaluation and other techniques. Minimization of temporary storage and the number of arithmetic registers in multiple register machines.

13. Optimization of loops. Typical loops coded several ways. Index register optimization in the innermost loop. Classification of loops for optimization purposes.

14. Problems of global optimization. Determination of flowchart graph of program. Analysis of program graphs. Rearrangement of computation to do as little as possible in innermost loop. Factoring of invariant subexpressions. Object code for interfaces between flow blocks.

ANNOTATED BIBLIOGRAPHY

1. ACM Compiler Symposium. Papers presented at the ACM Compiler Symposium, November 17–18, 1960, Washington, D. C., *Comm. ACM 4*, 1 (Jan. 1961), 3–84.

Contains a number of relevant papers including one by R. W. Floyd entitled "An Algorithm for Coding Efficient Arithmetic Operations" and one by P. Z. Ingerman on "Thunks."

2. Arden, B. W., Galler, B. A., and Graham, R. M. An algorithm for translating Boolean expressions. *J. ACM 9*, 2 (Apr. 1962), 222–239. CR-6341-3567.

Description of code generation in the Mad Compiler.

3. Brinch-Hansen, P., and House, R. The COBOL compiler for the Siemens 3003. *BIT 6*, 1 (1966), 1–23.

Describes the design of a ten-pass compiler with extensive error detection.

4. Cheatham, T. E., Jr. The TGS-II translator generator system. Proc. IFIP Congress, New York, 1965, Vol. 2, Spartan Books, New York, pp. 592–593.

A report on the "current position" of Computer Associates "translator generator system."

5. Cheatham, T. E., Jr. *The Theory and Construction of Compilers.* Document CA-6606-0111, Computer Associates, Inc., Wakefield, Mass., June 2, 1966, limited distribution.

Notes for course AM 295 at Harvard, fall 1967.

6. Cheatham, T. E., Jr., and Sattley, K. Syntax-directed compiling. Proc. AFIPS 1964 Spring Joint Comput. Conf., Vol. 25, Spartan Books, New York, pp. 31–57. CR-6455-6304.

An introduction to top-down syntax directed compilers.

7. Conway, M. E. Design of a separable transition-diagram compiler. *Comm. ACM 6*, 7 (July 1963), 396–408. CR-6451-5024.

Describes the organization of a Cobol compiler. The methods are largely applicable to construction of compilers for other languages such as Algol.

8. Conway, R. W., and Maxwell, W. L. CORC—the Cornell computing language. *Comm. ACM 6*, 6 (June 1963), 317–321.

Description of a language and compiler which are designed to provide extensive error diagnostics and other aids to the programmer.

9. Ershov, A. P. ALPHA—an automatic programming system of high efficiency. *J. ACM 13*, 1 (Jan. 1966), 17–24. CR-6673-9720.

Describes a compiler for a language which includes most of Algol as a subset. Several techniques for optimizing both the compiler and the object code are presented.

10. Ershov, A. P. *Programming Programme for the BESM computer*, transl. by M. Nadler. Pergamon Press, New York, 1959, 158 pp. CR-6235-2595.

One of the earliest works on compilers. Introduced the use of stacks and the removal of common subexpressions.

11. Ershov, A. P. On programming of arithmetic operations. *Comm. ACM 1*, 8 (Aug. 1958), 3–6 and 9 (Sept. 1958), 16.

Gives an algorithm for creating rough machine language instructions in pseudoform and then altering them into a more efficient form.

12. Freeman, D. N. Error correction in CORC. Proc. AFIPS 1964 Fall Joint Computer Conf., Vol. 26, Part I, Spartan Books, New York, pp. 15–34.

Discusses techniques of correcting errors in programs written in the Cornell computing language.

13. Garwick, J. V. GARGOYLE, a language for compiler writing. *Comm. ACM 7*, 1 (Jan. 1964), 16–20. CR-6453-5675.

Describes an Algol-like language which uses syntax-directed methods.

14. Gear, C. W. High speed compilation of efficient object code. *Comm. ACM 8*, 8 (Aug. 1965), 483–488. CR-6671-9000.

Describes a three-pass compiler which represents a compromise between compilation speed and object code efficiency. Primary attention is given to the optimization performed by the compiler.

15. Gries, D., Paul, M., and Wiehle, H. R. Some techniques used in the ALCOR ILLINOIS 7090. *Comm. ACM 8*, 8 (Aug. 1965), 496-500. CR-6566-8556.

Describes portions of an Algol compiler for the IBM 7090.

16. Hawkins, E. N., and Huxtable, D. H. R. A multipass translation scheme for ALGOL 60. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 3*, Pergamon Press, New York, 1963, pp. 163-206.

Discusses local and global optimization techniques.

17. Horwitz, L. P., Karp, R. M., Miller, R. E., and Winograd, S. Index register allocation. *J. ACM 13*, 1 (Jan. 1966), 43-61. CR-6674-10,068.

A mathematical treatment of the problem. Useful in compiler writing.

18. International Computation Centre (Eds.) *Symbolic Languages in Data Processing*, Proceedings of the Symposium in Rome, March 26-31, 1962. Gordon and Breach, New York, 1962, 849 pp.

The twelve papers listed under "Construction of Processors for Syntactically Highly Structured Languages" in this volume are particularly of interest for this course.

19. Knuth, D. E. A history of writing compilers. *Comput. Autom. 11*, 12 (Dec. 1962), 8-18.

Describes some of the early techniques used in writing American compilers.

20. McClure, R. M. TMG—a syntax directed compiler. Proc. ACM 20th Nat. Conf., 1965, Association for Computing Machinery, New York, pp. 262-274.

The compiler writing system described in this paper was designed to facilitate the construction of simple one-pass translators for some specialized languages. It has features which simplify the handling of declarative information and errors.

21. Naur, P. The design of the GIER ALGOL compiler. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 4*, Pergamon Press, New York, 1964, pp. 49-85. CR-6564-7904.

Describes a multipass compiler written for a computer with a small high-speed memory.

22. Randell, B., and Russell, L. J. *ALGOL 60 Implementation*. Academic Press, New York, 1964, 418 pp. CR-6565-8246.

Contains a survey of Algol implementation techniques and a description of an error-checking and debugging compiler for the KDF9 computer.

23. Reynolds, J. C. An introduction to the COGENT programming system. Proc. ACM 20th Nat. Conf., 1965, Association for Computing Machinery, New York, pp. 422-436.

Describes the structure and major facilities of a compiler-compiler system which couples the notion of syntax-directed compiling with that of recursive list processing.

24. Rosen, S. (Ed.) *Programming Systems and Languages*. McGraw-Hill, New York, 1967, 734 pp.

A collection of papers of which the following are of special interest for this course: Backus, et al., pp. 29-47; Bauer and Samelson, pp. 206-220; Dijkstra, pp. 221-227; Kanner, Kosinski, and Robinson, pp. 228-252; Rosen, Spurgeon, and Donnelly, pp. 264-297; and Rosen, pp. 306-331.

25. Schorre, D. V. META II, a syntax-oriented compiler writing language. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. D1.3-1 to D1.3-11. CR-6561-6943.

Describes a compiler writing language in which its own compiler can be written.

26. Wegner, P. (Ed.) *Introduction to System Programming*. Academic Press, New York, 1965, 316 pp. CR-6455-6300.

Contains a collection of papers of which the following are of special interest for this course: Pyle, pp. 86-100; Wegner, pp. 101-121; Randell, pp. 122-136; Huxtable, pp. 137-155; Hoare, pp. 156-165; and d'Agapeyeff, pp. 199-214.

Course I6. Switching Theory (3-0-3) or (2-2-3)

APPROACH

This course should present the theoretical foundations and mathematical techniques concerned with the design of logical circuits. Examples should be chosen to illustrate the applicability to computers or other digital systems whenever possible. Facility with Boolean algebra and appreciation for the effects of delays should be developed. Some laboratory experiments are highly desirable.

CONTENT

1. Review of nondecimal number systems. Introduction to unit-distance, error-correcting, and other codes.

2. Development of switching algebra and its relation to Boolean algebra and propositional logic. Brief discussion of switching elements and gates. Analysis of gate networks. Truth tables and completeness of connectives.

3. Simplification of combinational networks. Use of map and tabular techniques. The prime implicant theorem. Threshold logic.

4. Different modes of sequential circuit operation. Flow table, state diagram, and regular expression representations. Clocked circuits. Flip-flop and feedback realizations.

5. Synthesis of sequential circuits. State minimization and internal variable assignments for pulse and fundamental mode circuits. Race considerations. Iterative and symmetric networks.

6. Effects of delays. Static, dynamic, and essential hazards.

ANNOTATED BIBLIOGRAPHY

As is indicated, several of the books listed below could be used as texts for this course, but it probably would be desirable to supplement any of them with additional material.

General textbooks

1. Caldwell, S. H. *Switching Circuits and Logical Design*. Wiley, New York, 1958, 686 pp.

The classic book on relay-oriented switching theory.

2. Harrison, M. A. *Introduction to Switching and Automata Theory*. McGraw-Hill, New York, 1965, 499 pp. CR-6671-9109.

A mathematical and abstract reference for advanced topics.

3. Higonnet, R. A., and Grea, R. A. *Logical Design of Electrical Circuits*. McGraw-Hill, New York, 1958, 220 pp.

Almost exclusively devoted to relay networks.

4. Humphrey, W. S., Jr. *Switching Circuits with Computer Applications*. McGraw-Hill, New York, 1958, 264 pp.

A somewhat out-of-date undergraduate level text.

5. Krieger, M. *Basic Switching Circuit Theory*. Macmillan, New York, 1967, 256 pp. CR-6784-12,510.

Basic elementary treatment which does not discuss hazards or codes.

6. McCluskey, E. J., Jr. *Introduction to the Theory of Switching Circuits*. McGraw-Hill, New York, 1965, 318 pp. CR-6673-9834.

A possible text for this course.

7. McCluskey, E. J., Jr., and Bartee, T. C. (Eds.) *A Survey of Switching Circuit Theory*. McGraw-Hill, New York, 1962, 205 pp. CR-6342-3958.

A collection of papers. Weak as a text since there are no problems. Possibly of some value as reading to illustrate different approaches.

8. Maley, G. A., and Earle, J. *The Logic Design of Transistor Digital Computers*. Prentice-Hall, Englewood Cliffs, N. J., 1963, 322 pp. CR-6345-4582.

Despite its title, this book covers a considerable amount of switching theory. Emphasis is on NOR circuits and asynchronous systems, and on techniques rather than theorems. Numerous examples.

9. MARCUS, M. P. *Switching Circuits for Engineers.* Prentice-Hall, Englewood Cliffs, N. J., 1962, 296 pp. CR-6341-3681.
   Broad but not too mathematical coverage of switching theory.
10. MILLER, R. E. *Switching Theory, Vol. 1, Combinational Circuits.* Wiley, New York, 1965, 351 pp. CR-6565-8369.
   Highly mathematical and somewhat advanced for an undergraduate course. Interesting discussion of the effects of delays.
11. PRATHER, R. E. *Introduction to Switching Theory: A Mathematical Approach.* Allyn and Bacon, Boston, 1967, 496 pp.
   A highly mathematical and broad coverage of both combinatorial switching theory and sequential machine theory.
12. TORNG, H. C. *Introduction to the Logical Design of Switching Systems.* Addison-Wesley, Reading, Mass., 1965, 286 pp. CR-6456-6806.
   General elementary coverage including a discussion of switching elements and magnetic logic. Outmoded discussion of iterative (cascaded) networks. Many computer-related examples.
13. WARFIELD, J. N. *Principles of Logic Design.* Ginn and Co., Boston, 1963, 291 pp. CR-6451-5136.
   Covers some elementary switching theory in the context of computer logic.

More specialized books

14. CURTIS, V. A. *A New Approach to the Design of Switching Circuits.* D. Van Nostrand, Princeton, N. J., 1962, 635 pp. CR-6346-4818.
   Devoted mainly to decomposition theory for combinational circuits. Useful as a reference in this area and as a source of examples since it contains many detailed sample problems.
15. DERTOUZOS, M. L. *Threshold Logic: A Synthesis Approach.* M.I.T. Press, Cambridge, Mass., 1965, 256 pp. CR-6676-10,929.
   Concentrates on the characterization and application of threshold elements in terms of logical design.
16. HU, S. T. *Threshold Logic.* University of California Press, Berkeley, Calif., 1965, 338 pp.
   A comprehensive reference which also contains some of the author's original research.
17. LEWIS, P. M., AND COATES, C. L. *Threshold Logic.* Wiley, New York, 1967, 483 pp.
   Emphasizes single and multigate networks for controlled sensitivity.
18. PHISTER, M., JR. *Logical Design of Digital Computers.* Wiley, New York, 1958, 401 pp.
   Covers the application of sequential circuit theory to design of computer logic. Considers only clocked circuits and (for the most part) serial operation.

Course I7. Sequential Machines (3-0-3)

APPROACH

This is to be a rigorous theoretical course. The material is about evenly devoted to three major points of view: the structural aspects of sequential machines, the behavioral aspects of sequential machines, and the variants of finite automata. Machines with unbounded storage such as Turing machines and pushdown-store automata are to be covered in course A7.

CONTENT

1. Definition of finite automata and sequential machines. Various methods of representing automata including state tables, state diagrams, set theoretic methods, and sequential nets. (3 lectures)
2. Equivalent states and equivalent machines. Reduction of states in sequential machines. (3 lectures)
3. Right-invariant and congruence relations. The equivalence of nondeterministic and deterministic finite automata. Closure properties of languages definable by finite automata and the Kleene-Myhill theorem on regular languages. (4 lectures)
4. Decision problems of finite automata. Testing equivalence, acceptance of a nonempty set, acceptance of an infinite set. (3 lectures)
5. Incomplete sequential machines. Compatible states and algorithms for constructing minimal state incomplete sequential machines. (3 lectures)
6. The state assignment problem. Series-parallel decompositions. Coordinate assignments. (2 lectures)
7. Algebraic definition of a sequential machine. Homomorphisms of monoids. (2 lectures)
8. Partitions with the substitution property. Homomorphism decompositions into a series-composition. (2 lectures)
9. Decomposition of permutation automata. (1 lecture)
10. The decomposition of finite-state automata into a cascade of permutation-reset automata using the method of set systems (covers). (2 lectures)
11. Final series-parallel decomposition into a cascade of two-state automata and simple-group automata. (2 lectures)
12. Brief introduction to undecidability notions. The halting problem. (2 lectures)
13. Multitape nonwriting automata. (4 lectures)
14. Generalized sequential machines. (2 lectures)
15. Subsets of regular languages. Group-free automata. (3 lectures)
16. Regular expressions. Algebra, derivatives and star height. (5 lectures)
17. Probabilistic automata. (3 lectures)

ANNOTATED BIBLIOGRAPHY

Except for two survey articles, only books are included in the following list. Since this field has developed within the last fifteen years, much of the material is still in the periodical literature.

1. CAIANIELLO, E. R. (Ed.) *Automata Theory.* Academic Press, New York, 1966, 343 pp. CR-6676-10,935.
   A collection of research and tutorial papers on automata, formal languages, graph theory, logic, algorithms, recursive function theory, and neural nets, which, because of varying interest and difficulty, might be useful for supplementary reading by ambitious students.
2. FISCHER, P. C. Multitape and infinite-state automata—a survey. *Comm. ACM 8*, 12 (Dec. 1965), 799–805. CR-6675-10,561.
   A survey of machines which are more powerful than finite automata and less powerful than Turing machines. Also an extensive bibliography.
3. GILL, A. *Introduction to the Theory of Finite-State Machines.* McGraw-Hill, New York, 1962, 207 pp. CR-6343-4207.
   An automata theory approach to finite-state machines which is somewhat engineering oriented and written at a fairly elementary level.
4. GINSBURG, S. *An Introduction to Mathematical Machine Theory.* Addison-Wesley, Reading, Mass., 1962, 137 pp. CR-6452-5431.
   A text on the behavior of the sequential machines of Huffman-Moore-Mealy, abstract machines of Ginsburg, and tape recognition devices of Rabin and Scott.
5. GLUSHKOV, V. M. *Introduction to Cybernetics*, transl. by Scripta Technica, Inc. Academic Press, New York, 1966, 324 pp.
   A translation of the Russian text which assumes only a limited background. Approaches subject from somewhat different point of view than most Western texts. Contains much relevant material.
6. HARRISON, M. *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965, 499 pp. CR-6671-9109.
   This text for engineers and mathematicians develops the foundations of both switching and automata theory in abstract

mathematical terms. Emphasis is on switching theory. Coverage includes sequential machines, regular events, definite events, probabilistic machines, and context-free languages.

7. Hartmanis, J., and Stearns, R. E. *Algebraic Structure Theory of Sequential Machines.* Prentice-Hall, Englewood Cliffs, N. J., 1966, 211 pp. CR-6782-11,635.

   The first thorough treatment of the structure theory of sequential machines and its applications to machine synthesis and machine decomposition. A research monograph selected from a series of papers by the authors and not written as a text. Practically no exercises.

8. Hennie, F. C., III. *Iterative Arrays of Logical Circuits.* M.I.T. Press, Cambridge, Mass., and Wiley, New York, 1961, 242 pp. CR-6232-1733.

   Currently the most complete treatise on iterative arrays.

9. Kautz, W. H. (Ed.) *Linear Sequential Switching Circuits—Selected Technical Papers.* Holden-Day, San Francisco, 1965, 234 pp. CR-6674-10,205.

   A collection of papers on linear sequential machines.

10. McNaughton, R. The theory of automata—a survey. In F. L. Alt (Ed.), *Advances in Computing, Vol. 2,* Academic Press, New York, 1961, pp. 379–421. CR-6342-3920.

    Most of the areas of automata theory are included with the exception of switching theory and other engineering topics.

11. Miller, R. E. *Switching Theory, Vol. 2, Sequential Circuits and Machines.* Wiley, New York, 1965, 250 pp. CR-6783-12,120.

    Highly mathematical and somewhat advanced as a text for an undergraduate course.

12. Minsky, M. *Computation: Finite and Infinite Machines.* Prentice-Hall, Englewood Cliffs, N. J., 1967, 317 pp.

    The concept of an "effective procedure" is developed in this text. Also treats algorithms, Post productions, regular expressions, computability, infinite and finite-state models of digital computers, and computer languages.

13. Moore, E. F. (Ed.) *Sequential Machines: Selected Papers.* Addison-Wesley, Reading, Mass., 1964, 266 pp.

    This collection of classical papers on sequential machines includes an extensive bibliography by the editor.

14. Prather, R. E. *Introduction to Switching Theory: A Mathematical Approach.* Allyn and Bacon, Boston, 1967, 496 pp.

    A mathematical and broad treatment of both combinatorial switching theory and sequential machines.

15. Shannon, C. E., and McCarthy, J. (Eds.) *Automata Studies.* Princeton University Press, Princeton, N. J., 1956, 285 pp.

    A collection of many of the early papers on finite automata, Turing machines, and synthesis of automata which stimulated the development of automata theory. Philosophical papers, in addition to mathematical papers, are included, since the aim of the collection is to help explain the workings of the human mind.

Courses I8 and I9. Numerical Analysis I and II (3-0-3) and (3-0-3)

APPROACH

The numerical methods presented in these courses are to be developed and evaluated from the standpoint of efficiency, accuracy, and suitability for high-speed digital computing. While other arrangements of the material in these courses are possible, the ones suggested here do allow the two courses to be taught independently of one another.

CONTENT OF COURSE I8

1. *Solution of Equations.* Newton's method and other iterative methods for solving systems of equations. Aitken's δ^2 process. Newton-Bairstow method. Muller's method and Bernoulli's method for polynomial equations. Convergence conditions and rates of convergence for each method.

2. *Interpolation and Approximation.* Polynomial interpolation. Lagrange's method with error formula. Gregory-Newton and other equal interval interpolation methods. Systems of orthogonal polynomials. Least-squares approximation. Trigonometric approximation. Chebyshev approximation.

3. *Numerical Differentiation and Quadrature.* Formulas involving equal intervals. Romberg integration. Extrapolation to the limit. Gaussian quadrature.

4. *Solution of Ordinary Differential Equations.* Runge-Kutta methods. Multistep methods. Predictor-corrector methods. Stability.

CONTENT OF COURSE I9

5. *Linear Algebra.* Rigorous treatment of elimination methods and their use to solve linear systems, invert matrices, and evaluate determinants. Compact schemes. Methods for solving the eigenvalue-eigenvector problem including the power method, the inverse power method, Jacobi's method, Givens' method and Householder's method. Roundoff analysis and conditioning.

6. *Numerical Solution of Boundary Value Problems in Ordinary Differential Equations.*

7. *Introduction to the Numerical Solution of Partial Differential Equations.* Computational aspects of finite difference methods for linear equations. Determination of grids. Derivation of difference equations. Solution of large linear systems by iterative methods such as simultaneous displacements, successive displacements, and successive overrelaxation.

ANNOTATED BIBLIOGRAPHY

Listed below are some but by no means all of the books which could be used as texts and/or references for these courses. The more general texts normally include solution of polynomial and other nonlinear equations; interpolation, numerical quadrature, and numerical differentiation; ordinary differential equations; and linear algebra. Significant deviations from these are indicated by the annotations.

Besides listing books which might be used as texts for part or all of these courses, the following includes books for those desiring to go deeper into the various areas. In particular, Refs. 1, 3, 10, 17, and 18 have been included for linear algebra; Refs. 2 and 16 for partial differential equations; Ref. 15 for the solution of nonlinear equations; and Ref. 7 for ordinary differential equations.

1. Faddeev, D. K., and Faddeeva, V. N. *Computational Methods of Linear Algebra,* transl. by R. C. Williams. W. H. Freeman, San Francisco, 1963, 621 pp. CR-6016-0374.

   An excellent reference on the theory of computational methods in linear algebra. Does not treat the theory of computational errors. Introductory chapter could serve as a text for a course in linear algebra. Beginning analysis and an elementary knowledge of complex variables is assumed. Examples but no exercises.

2. Forsythe, G. E., and Wasow, W. R. *Finite-Difference Methods for Partial Differential Equations.* Wiley, New York, 1960, 444 pp.

   A fundamental reference on the numerical solution of partial differential equations by finite-difference methods. Provides a thorough treatment of hyperbolic, parabolic, and elliptic equations. Orientation is toward the use of high-speed computers, but it is not intended as a guide for programmers. For most of the book, advanced calculus and linear algebra provide sufficient background. Previous knowledge of partial differential equations not required. Some illustrative examples but no exercises.

3. Fox, L. *An Introduction to Numerical Linear Algebra.* Oxford University Press, New York, 1964, 295 pp. CR-6456-6723.

   A basic reference on computational methods in linear algebra. Designed for engineers and scientists as well as mathematicians. Emphasis on the principles involved rather than the details of applications to computers. Intended to prepare the reader for a more advanced book such as Wilkinson's *The Algebraic Eigenvalue Problem.* Introductory chapter on matrix algebra. Illustrative examples and exercises.

4. Fröberg, C. E. *Introduction to Numerical Analysis*. Addison-Wesley, Reading, Mass., 1965, 340 pp. CR-6671-9037.

Designed as a text for an undergraduate numerical analysis course. Includes, in addition to the standard topics, partial differential equations (briefly), approximation by Chebyshev polynomials and other functions, Monte Carlo methods, and linear programming. Emphasis on modern methods well-adapted for computers. Mathematically rigorous treatment with detailed error analysis given in typical cases. Presupposes differential and integral calculus and differential equations. Illustrative examples and exercises.

5. Hamming, R. W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, New York, 1962, 411 pp. CR-6236-3367.

Excellent as a reference. Provides interesting and different point of view. Treats interpolation and approximation; numerical differentiation and integration; and ordinary differential equations by polynomial and other methods such as Fourier methods, and exponentials. Brief treatments of nonlinear equations and linear algebra, simulation, and Monte Carlo methods. Presupposes beginning analysis, Fourier series, mathematical statistics, feed-back circuits, noise theory. Illustrative examples and exercises.

6. Henrici, P. *Elements of Numerical Analysis*. Wiley, New York, 1964, 328 pp.

Designed as a text for a one-semester course in numerical analysis. Covers the standard topics except linear algebra. Emphasis on numerical analysis as a mathematical discipline. A distinction is made between algorithms and theorems. Introductory chapters on complex variables and difference equations. Beginning analysis (12 semester hours) and ordinary differential equations are assumed. Illustrative examples and exercises.

7. Henrici, P. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley, New York, 1962, 407 pp. CR-6341-3733.

A basic reference on the numerical methods for solving ordinary differential equations. Designed as a text for a senior-level course on ordinary differential equations. Includes a mathematically rigorous treatment of various methods. Emphasis is on the study of discretization errors and round-off errors. Presupposes differential equations, advanced calculus, linear algebra, and elementary complex variables (though large parts of the book do not require all of these topics). Illustrative examples and exercises.

8. Hildebrand, F. B. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 1956, 511 pp.

A good book for supplementary reading though written in 1956. Gives primary emphasis to methods adapted for desk calculators. Includes standard topics except for linear algebra. Separate chapters on least-squares, polynomial approximation, Gaussian quadrature, and approximation of various types. Beginning analysis sufficient background for most of the book. An extensive set of exercises.

9. Householder, A. S. *Principles of Numerical Analysis*. McGraw-Hill, New York, 1953, 274 pp.

Good for supplementary reading. Designed as mathematical textbook rather than a compendium of computational rules. Published in 1953, the book includes many methods applicable only to hand computation though it was written with computers in mind. Includes the standard topics except ordinary differential equations. Presupposes beginning analysis plus some knowledge of probability and statistics. Some exercises.

10. Householder, A. S. *The Theory of Matrices in Numerical Analysis*. Blaisdell, New York, 1964, 257 pp.

Good for supplementary reading. Considers the development and appraisal of computational methods in linear algebra from the theoretical point of view. Does not develop specific computer flowcharts or programs. Presupposes a knowledge of matrix algebra. Illustrative examples and exercises.

11. Isaacson, E., and Keller, H. B. *Analysis of Numerical Methods*. Wiley, New York, 1966, 541 pp. CR-6783-11,966.

A very well written and rather comprehensive text presenting a careful analysis of numerous important numerical methods with a view toward their applicability to computers. With an appropriate selection of material the book lends itself well to use as a text; otherwise, it is an excellent reference.

12. Milne, W. E. *Numerical Solution of Differential Equations*. Wiley, New York, 1953, 275 pp.

Since it was written in 1953, much of this material has been superseded by more recent work; yet it remains very suitable for supplemental reading. Ordinary and partial differential equations are treated as well as some problems in linear algebra. Many of the methods are adapted for hand computation rather than for computers. Beginning analysis should provide sufficient background for most of the book. Illustrative examples and some exercises.

13. Ralston, A. *A First Course in Numerical Analysis*. McGraw-Hill, New York, 1965, 578 pp. CR-6671-9035.

Designed as a text for a one-year course in numerical analysis (though not all of the material could be covered) to be taken by graduate students and advanced undergraduate students, primarily in mathematics. Although numerical analysis is treated as a full-fledged branch of applied mathematics, orientation is toward the use of digital computers. Basic topics in numerical analysis covered thoroughly. Separate chapters devoted to functional approximation by least-squares techniques and by minimum-maximum error techniques. Presupposes beginning analysis, advanced calculus, orthogonal polynomials, and complex variables. A course in linear algebra is assumed for the chapters in that area. An extensive set of illustrative examples and exercises.

14. Todd, J. (Ed.) *A Survey of Numerical Analysis*. McGraw-Hill, New York, 1962, 589 pp. CR-6236-3368.

Written by a number of authors. Some of the early chapters have been used in connection with introductory courses. Because of its breadth of coverage it is especially suited as a reference for these courses. Besides the usual topics, there are separate chapters on orthogonalizing codes, partial differential equations, integral equations, and problems in number theory. The prerequisites vary with the chapters but for early chapters beginning analysis and linear algebra would suffice. Exercises given in some of the early chapters.

15. Traub, J. F. *Iterative Methods for the Solution of Equations*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 310 pp. CR-6672-9339.

A good reference on the numerical solution of equations and (briefly) systems of equations by iteration algorithms. The methods are treated with rigor, though rigor in itself is not the main object. Contains a considerable amount of new material. Many illustrative examples.

16. Varga, Richard. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N. J., 1962, 322 pp. CR-6343-4236.

An excellent reference giving theoretical basis behind methods for solving large systems of linear algebraic equations which arise in the numerical solution of partial differential equations by finite-difference methods. Designed as a text for a first-year graduate course in mathematics.

17. Wilkinson, J. H. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, N. J., 1964, 161 pp. CR-6455-6341.

Studies the cumulative effect of rounding errors in computations involving large numbers of arithmetic operations performed by digital computers. Special attention given to problems involving polynomials and matrices. A very important reference for a computer-oriented course in numerical analysis.

18. Wilkinson, J. H. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965, 662 pp.

A basic reference on computational methods in linear algebra. Provides a thorough treatment of those methods with which the author has had direct numerical experience on the computer. Treats the methods theoretically and also from the standpoint of rounding errors. Presupposes beginning analysis, linear algebra, and elementary complex variables. Illustrative examples but no exercises.

Course A1. Formal Languages and Syntactic Analysis (3-0-3)

APPROACH

This course combines the theoretical concepts which arise in formal language theory with their practical application to the syntactic analysis of programming languages. The objective is to build a bridge between theory and practical applications, so that the mathematical theory of context-free languages becomes meaningful to the programmer and the theoretically oriented student develops an understanding of practical applications. Assignments in this course should include both computer programming assignments and theorem proving assignments.

CONTENT

The following topics should be covered, but the organization of the material and relative emphasis on individual topics is subject to individual preference.

1. Definition of a formal grammar as notation for specifying a subset of the set of all strings over a finite alphabet, and of a formal language as a set specified by a formal grammar. Production notation for specifying grammars. Recursively enumerable, context-sensitive, context-free, and finite-state grammars. Examples of languages specified by grammars such as a^nb^n, $a^nb^nc^n$.
2. Specification of arithmetic expressions and arithmetic statements as context-free grammars. Use of context-free grammars as recognizers. Use of recognizers as a component in compilation or interpretive execution of arithmetic statements.
3. Syntactic analysis, recognizers, analyzers and generators. Top-down and bottom-up algorithms. The backtracking problem and the reduction of backtracking by bounded context techniques. Theory of bounded context analysis and LR(k) grammars.
4. Precedence and operator precedence techniques. The algorithms of Floyd, Wirth and Weber, etc. Semantics of precedence grammars for arithmetic statements and simple block structure.
5. Top-down and bottom-up algorithms for context-free languages. The algorithms of Cheatham, Domolki, and others.
6. Languages for syntactic analysis and compilation such as COGENT and TMG. A simple syntactic compiler written in one of these languages.
7. Reductive grammars, Floyd productions, and semantics for arithmetic expressions. The work of Evans, Feldman, and others.
8. Theory of context-free grammars, normal forms for context-free grammars, elimination of productions of length zero and one. Chomsky and Greibach normal forms. Ambiguous and inherently ambiguous grammars. The characteristic sequence of a grammar. Strong equivalence, weak equivalence and equivalence with preservation of ambiguity. Asymptotic time and space requirements for context-free language recognition.
9. Combinatorial theorems for context-free grammars. Proof that $a^nb^nc^n$ cannot be represented by a context-free grammar. Linear and semilinear sets. Parikh's theorem.
10. Grammars and mechanical devices. Turing machines, linear bounded automata, pushdown automata, finite-state automata, and the corresponding grammars.
11. Properties of pushdown automata. Deterministic and nondeterministic automata and languages. Stack automata. Programming languages and pushdown automata.

ANNOTATED BIBLIOGRAPHY

1. BAR-HILLEL, Y., PERLES, M., AND SHAMIR, E. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung 14* (1961), 143–172. (Reprinted in Y. Bar-Hillel (Ed.), *Languages and Information, Selected Essays.* Addison-Wesley, Reading, Mass., 1964. CR-6562-7178.)
   A well-written paper containing the first statement of many of the principal results of context-free languages.
2. BAUER, F. L., AND SAMELSON, K. Sequential formula translation. *Comm. ACM 3*, 2 (Feb. 1960), 76–83. CR-6015-0219.
   The first systematic paper on the translation of programming languages from left to right using precedence techniques.
3. BROOKER, R. A., AND MORRIS, D. A general translation program for phrase structure grammars. *J. ACM 9*, 1 (Jan. 1962), 1–10.
   Summarizes the design and machine-oriented characteristics of a syntax-directed compiler and describes both its syntactic and semantic features.
4. CHEATHAM, T. E., JR., AND SATTLEY, K. Syntax directed compiling. Proc. AFIPS 1964 Spring Joint Comput. Conf., Vol. 25, Spartan Books, New York, pp. 31–57. CR-6455-6304.
   A description of one of the earliest operational top-down syntax-directed compilers.
5. CHOMSKY, N. Formal properties of grammars. In R. R. Bush, E. H. Galanter, and R. D. Luce (Eds.), *Handbook of Mathematical Psychology, Vol. 2*, Wiley, New York, 1962, pp. 323–418. CR-6676-10,731.
   An excellent review of the work of both Chomsky and others in this field. Contains a good bibliography. See also the two companion chapters in this volume written by Chomsky and G. A. Miller.
6. EVANS, A. An ALGOL 60 compiler. In R. Goodman (Ed.), *Annual Review in Automatic Programming, Vol. 4*, Pergamon Press, New York, 1964, pp. 87–124. CR-6564-7905.
   The first compiler design application of a reductive syntax with labelled productions.
7. FELDMAN, J. A. A formal semantics for computer languages and its application in a compiler-compiler. *Comm. ACM 9*, 1 (Jan. 1966), 3–9. CR-6674-10,080.
   A description of a formal semantic language which can be used in conjunction with a language for describing syntax to specify a syntax-directed compiler.
8. FLOYD, R. W. A descriptive language for symbol manipulation. *J. ACM 8*, 4 (Oct. 1961), 579–584. CR-6234-2140.
   The first discussion of reductive grammars, including a reductive grammar from which the first example in Ref. 9 was derived. The association of semantics with syntactic recognition is directly illustrated.
9. FLOYD, R. W. Syntactic analysis and operator precedence. *J. ACM 10*, 3 (July 1963), 316–333.
   Defines the notions of operator grammars, precedence grammars (here called "operator precedence grammars"), precedence functions, and a number of other concepts. Examples of precedence grammars and nonprecedence grammars.
10. FLOYD, R. W. Bounded context syntactic analysis. *Comm. ACM 7*, 2 (Feb. 1964), 62–67. CR-6454-6074.
    Introduces the basic concepts of bounded context grammars and gives a set of conditions for testing whether a given grammar is of bounded context (m, n).
11. FLOYD, R. W. The syntax of programming languages—a survey. *IEEE Trans. EC-13*, 4 (Aug. 1964), 346–353.
    An expository paper which defines and explains such concepts as phrase-structure grammars, context-free languages, and syntax-directed analysis. Extensive bibliography.
12. GINSBURG, S. *The Mathematical Theory of Context-Free Languages.* McGraw-Hill, New York, 1966, 232 pp. CR-6783-12,074.
    Presents a rigorous discussion of the theory of context-free languages and pushdown automata.
13. GREIBACH, S. A. A new normal-form theorem for context-free phrase structure grammars. *J. ACM 12*, 1 (Jan. 1965), 42–52. CR-6564-7830.
    Shows that every grammar is equivalent with preservation of ambiguities to a grammar in the "Greibach Normal Form."
14. GRIFFITHS, T. V., AND PETRICK, S. R. On the relative efficiencies of context-free grammar recognizers. *Comm. ACM 8*, 5 (May 1965), 289–300. CR-6564-7999.

A comparative discussion of a number of syntactic analysis algorithms.

15. IEEE Computer Group, Switching and Automata Theory Committee. Conf. Rec. 1967 8th Ann. Symposium on Switching and Automata Theory. Special Publication 16 C 56, Institute of Electrical and Electronic Engineers, New York, 1967.

Contains a number of papers relevant to this course including those by: Rosenkrantz, pp. 14–20; Aho, pp. 21–31; Hopcroft and Ullman, pp. 37–44; Ginsburg and Greibach, pp. 128–139.

16. Irons, E. T. A syntax directed compiler for ALGOL 60. *Comm. ACM 4*, 1 (Jan. 1961), 51–55.

The first paper on syntactic compilation. It discusses both a top-down implementation of a syntactic compiler and the way in which semantics is associated with the generation steps of such a compiler.

17. Irons, E. T. Structural connections in formal languages. *Comm. ACM 7*, 2 (Feb. 1964), 67–72. CR-6455-6212.

The concept of structural connectedness defined is essentially the same as that of a bounded context grammar.

18. Knuth, D. E. On the translation of languages from left to right. *Inf. Contr. 8*, 6 (Dec. 1965), 607–639. CR-6674-10,162.

The concept of a grammar which permits translation from left to right with forward context *k* (LR(*k*) grammar) is developed and analyzed.

19. McClure, R. M. TMG—a syntax directed compiler. Proc. ACM 20th Nat. Conf., 1965, pp. 262–274.

A description of a compiler in which the syntax is specified by an ordered sequence of labeled productions and in which semantics can be explicitly associated with productions.

20. Naur, P. (Ed.) Revised report on the algorithmic language, ALGOL 60. *Comm. ACM 6*, 1 (Jan. 1963), 1–17. CR-6345-4540.

The first systematic application of context-free languages to the description of actual programming languages.

21. Reynolds, J. C. An introduction to the COGENT programming system. Proc. ACM 20th Nat. Conf., 1965, pp. 422–436.

Describes the structure and major facilities of a compiler-compiler system which couples the notion of syntax-directed compiling with that of recursive list processing.

22. Wirth, N., and Weber, H. EULER: a generalization of ALGOL, and its formal definition, Parts I & II. *Comm. ACM 9*, 1 (Jan. 1966), 13–23, and 2 (Feb. 1966), 89–99.

The first discussion of pure precedence grammars and their use in Algol-like languages. Many of the concepts introduced by other authors are discussed in an illuminating way.

Course A2. Advanced Computer Organization (3-0-3)

APPROACH

This title could label either a course in the organization of advanced computers or an advanced course in computer organization. It is meant to be primarily the latter, with some material on novel computer organizations included. The approach is that of "comparative anatomy": first, each of several organization and system design problems should be identified; then, a comparison of the solutions to the problem should be made for several different computers; next, the rationale of each solution should be discussed; and finally, an attempt should be made to identify the best solution. Students should prepare papers on designs used in other machines for several of the problem areas. These papers should include discussions of the circuit technology available at the time the machine was designed and the intended market for the machine, and they should compare the machine design with other designs.

CONTENT

The computer system design problem areas which should be covered in this course are listed below followed by a list of some of the machines which might be used to illustrate various solutions to these problems. Each major problem area should be discussed in general and at least three actual machine designs should be used to illustrate the widest possible range of solutions. A brief overall system description of each machine should be given before it is used to illustrate a particular design area.

Computer System Design Problem Areas

1. *Arithmetic Processing.* Integer and floating point representation, round-off and truncation, word and register lengths. Number and types of arithmetic units, malfunction detection and reaction, arithmetic abort detection, and reaction (overflow, etc.).
2. *Nonarithmetic Processing.* Addressable quanta and operation codes. Compatibility and interaction of nonarithmetic and arithmetic operands. Types of additional processing units.
3. *Memory Utilization.* Relationship between memory width and addressable quanta, memory block autonomy and phasing, memory access priorities (operands, instructions, input-output, etc.). Factoring of command-fetch processes (look-ahead).
4. *Storage Management.* Relocation, paging, and renaming. Storage protection. Hierarchy storage provisions and transfer mechanisms.
5. *Addressing.* Absolute addressing, indexing, indirect addressing, relative addressing, and base addressing.
6. *Control.* Clocking, interrupt processing, privileged mode operations, and autonomy of control functions.
7. *Input-Output.* Buffer facilities, channels (autonomy, interaction, interrupts), processing options (editing, formatting), input-output byte size versus memory width versus addressing quanta. Rate-matching (especially for input-output devices with inertia) and channel-sharing.
8. *Special System Designs.* Array or cellular computers, variable structure computers, and other advanced designs.

Illustrative Computers

ATLAS	one of the first machines to use paging
Bendix G-15	a bit serial machine with drum store
Burroughs B5000	a zero-address machine with stacks and Polish processing
CDC 6600	a very high-speed computer using look-ahead, instruction stacking and multiple peripheral processors
GIER (Denmark)	a machine which uses a small core storage with an auxiliary drum
IBM 701	a classic Von Neumann binary machine
IBM 1401	a serial character machine with peculiar addressing
IBM STRETCH	a machine which was intended to incorporate all state-of-the-art knowledge
IBM 360/ij	a series of machines which combine character and word handling capabilities
KDF-9	a computer using a nesting store
PB 440	a microprogrammable computer
SDS Sigma 7	a real-time time-sharing computer
UNIVAC I	a classic decimal machine

ANNOTATED BIBLIOGRAPHY

In addition to the references cited below, it is important that a collection of reference manuals for the actual computers be available to the student. These can be obtained from the computer manufacturers in most cases or from various reports such as the *Auerbach Standard EDP Reports*. In some cases where the title of the citation is self-explanatory, no annotation is given.

General references

1. Amdahl, G. M., Blaauw, G. A., and Brooks, F. P., Jr. Architecture of the IBM System/360. *IBM J. Res. Develop. 8*, 2 (Apr. 1964), 87–101. CR-6465-8374.

Although not a technical paper, it does give some insight into the decisions which determined many of the features of this family of computers.

2. Blaauw, G. A., et al. The structure of System/360. *IBM Syst. J. 3*, 2 (1964), 119–195.

   Describes the design considerations relating to the implementation, performance, and programming of the System/360 family of computers.

3. Boutwell, E. O., Jr., and Hoskinson, E. A. The logical organization of the PB 440 microprogrammable computer. Proc. AFIPS 1963 Fall Joint Comput. Conf., Vol. 24, Spartan Books, New York, pp. 201–213.

   Describes the use of a fast-read, slow-write memory for microprograms. The bus structure connecting the processing registers allows data transfers under control of the microprogram.

4. Buchholz, W. The system design of the IBM 701 computer. *Proc. IRE 41*, 10 (Oct. 1953), 1262–1274.

   Describes one of the first commercial binary computers.

5. Buchholz, W. (Ed.) *Planning A Computer System.* McGraw-Hill, New York, 1962, 336 pp. CR-6346-4786.

   Describes in reasonable detail a design philosophy and its bearing on specific design decisions for an entire computer system—in this case, Stretch.

6. Devonald, C. H., and Fotheringham, J. A. The ATLAS computer. *Datamation 7*, 5 (May 1961), 23–27. CR-6231-1405.

7. Digital Computer Laboratory, University of Illinois. *On the Design of a Very High Speed Computer.* Rep. No. 80, U. of Illinois, Urbana, Ill., 1957.

   A description of the first design for Illiac II.

8. Eckert, J. P., et al. The UNIVAC system. *Rev. of Elec. Dig. Comput.*, Proc. Joint IRE-AIEE Conf., Philadelphia, Pa., 10–12 Dec. 1951, pp. 6–14.

   A description of the Univac I computer.

9. Engineering Summer Conference, University of Michigan. Theory of computing machine design (course notes). U. of Michigan, Ann Arbor, Mich., 1960–1962. (Distribution of these notes was limited to participants.)

   Cover many aspects of design—from the applications of automata theory to the system design of parallel computers.

10. Fernbach, S. Very high-speed computers, 1964—the manufacturers' point of view. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Pt. II, Spartan Books, New York, 1965, pp. 33–141.

    Contains detailed reports on the CDC 6600, the IBM System/360 Model 92, and a Philco multiprocessing system.

11. Gram, C., et al. GIER—a Danish computer of medium size. *IEEE Trans. EC-12*, 6 (Dec. 1963), 629–650.

    Gives an evaluation of the order structure and the hardware organization and describes the operating system and the Algol 60 system.

12. Gray, H. J. *Digital Computer Engineering.* McGraw-Hill, New York, 1963, 381 pp. CR-6341-3654.

    Chap. 1 discusses Eniac and Edvac as examples of parallel and serial organization. Chaps. 2–4 deal with organization problems.

13. Gschwind, H. W. *Design of Digital Computers.* Springer-Verlag, New York, 1967, 530 pp.

    A general reference.

14. Hellerman, H. *Digital Computer System Principles.* McGraw-Hill, New York, 1967, 424 pp.

    A possible text. A good reference.

15. Hollander, G. L. (Ch.) The best approach to large computing capacity—a debate. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 463–485.

    Presents four approaches to achieving large computing capacity through: aggregation of conventional system elements (G. P. West); associative parallel processing (R. H. Fuller); an array computer (D. L. Slotnick); and the single-processor approach (G. M. Amdahl).

16. Mendelson, M. J., and England, A. W. The SDS Sigma 7: a real-time time-sharing system. Proc. AFIPS 1966 Fall Joint Comput. Conf., Vol. 29, Spartan Books, New York, pp. 51–64. CR-6700-0752.

    Discusses seven critical design problems—including interrupt processing, memory protection, space sharing, and recursive processing—and their solutions.

17. Richards, R. K. *Electronic Digital Systems.* Wiley, New York, 1966, 637 pp. CR-6676-10,649.

    Contains a good discussion of reliability and design automation.

18. Walz, R. F. Digital computers—general purpose and DDA. *Instrum. and Automat. 28*, 9 (Sept. 1955), 1516–1522.

    Describes the G-15 computer and the use of a digital differential analyzer (DDA) in general purpose digital systems.

19. Ware, W. H. *Digital Computer Technology and Design: Vol. I, Mathematical Topics, Principles of Operation and Programming; Vol. II, Circuits and Machine Design.* Wiley, New York, 1963, 237 pp. and 521 pp. CR-6562-7103 and 7104.

    Useful because of its coverage of early design techniques. Contains extensive bibliographies (at the end of each chapter), which refer to papers presenting the design features of numerous machines.

References on arithmetic and control

20. Blaauw, G. A. Indexing and control-word techniques. *IBM J. Res. Develop. 3*, 3 (July 1959), 288–301.

    Describes some of the control techniques used in the Stretch computer.

21. Beckman, F. S., Brooks, F. P., Jr., and Lawless, W. J., Jr. Developments in the logical organization of computer arithmetic and control units. *Proc. IRE 49*, 1 (Jan. 1961), 53–66. CR-6232-1680.

    Summarizes the developments in logical design and in arithmetic and control units through 1960. Contains a good bibliography.

22. Brooks, F. P., Jr., Blaauw, G. A., and Buchholz, W. Processing data in bits and pieces. *IEEE Trans. EC-8*, 3 (June 1959), 118–124. CR-6012-0035.

    Describes a data-handling unit which permits variable length binary or decimal arithmetic.

23. Sumner, F. H. The central control unit of the ATLAS computer. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 292–296. CR-6342-3961.

References on storage management

24. Arden, B. W., Galler, B. A., O'Brien, T. C., and Westervelt, F. H. Program and addressing structure in a time-sharing environment. *J. ACM 13*, 1 (Jan. 1966), 1–16. CR-6781-11,210.

    Describes the hardware and software devices used to facilitate program switching and efficient use of storage in a time-sharing computer system.

25. Belady, L. A. A study of replacement algorithms for a virtual memory computer. *IBM Syst. J. 5*, 2 (1966), 78–101.

    Discusses several algorithms for automatic memory allocation and compares them using the results of several simulation runs.

26. Cocke, J., and Kolsky, H. G. The virtual memory in the STRETCH computer. Proc. AFIPS 1959 Eastern Joint Comput. Conf., Vol. 16, Spartan Books, New York, pp. 82–93.

27. Evans, D. C., and Leclerk, J. Y. Address mapping and the control of access in an interactive computer. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 23–30.

    Describes the hardware implementation of a design based on separate program and data entities.

28. Gibson, D. H. Considerations in block-oriented systems design. Proc. AFIPS 1967 Spring Joint Comput. Conf., Vol. 30, Thompson Book Co., Washington, D. C., pp. 75–80.

    Analyzes block size, high-speed storage requirements, and job mix as they affect system design.

Reference on stack computers

29. Allmark, R. H., and Lucking, J. R. Design of an arithmetic unit incorporating a nesting store. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 694–698. CR-6455-6460.

    Describes the arithmetic unit for the KDF-9 computer.

30. Haley, A. C. D. The KDF-9 computer system. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22, Spartan Books, New York, pp. 108–120. CR-6452-5466.

    Describes the stack register concept, the means used to communicate with it, and the use of zero-address instructions of variable length.

31. Barton, R. S. A new approach to the functional design of a digital computer. Proc. AFIPS 1961 Western Joint Comput. Conf., Vol. 19, Spartan Books, New York, pp. 393–396. CR-6234-2158.

    The earliest published description of a Polish string processor—the B5000.

Parallel and variable structure organizations

32. Estrin, G. Organization of computer systems: the fixed plus variable structure computer. Proc. AFIPS 1960 Western Joint Comput. Conf., Vol. 17, Spartan Books, New York, pp. 33–40. CR-6235-2643.

33. Estrin, G., and Viswanathan, C. R. Organization of a fixed plus variable structure computer for computation of eigenvalues and eigenvectors of real symmetric matrices. *J. ACM 9*, 1 (Jan. 1962), 41–60.

34. Estrin, G., and Turn, R. Automatic assignment of computations in a variable structure computer system. *IEEE Trans. EC-12*, 6 (Dec. 1963), 755–773.

35. Gregory, J., and McReynolds, R. The SOLOMON computer. *IEEE Trans. EC-12*, 6 (Dec. 1963), 774–781.

    Presents the system organization, functional description, and circuit design from a total systems viewpoint.

36. Holland, J. H. A universal computer capable of executing an arbitrary number of subprograms simultaneously. Proc. AFIPS 1959 Eastern Joint Comput. Conf., Vol. 16, Spartan Books, New York, pp. 108–113.

37. Holland, J. H. Iterative circuit computers. Proc. AFIPS 1960 Western Joint Comput. Conf., Vol. 17, Spartan Books, New York, pp. 259–265.

38. Slotnick, D. L., Borck, W. C., and McReynolds, R. C. The SOLOMON computer. Proc. AFIPS 1962 Fall Joint Comput. Conf., Vol. 22, Spartan Books, New York, pp. 97–107.

    A general description of the philosophy and organization of a highly parallel computer design which is a predecessor of Illiac IV.

39. Schwartz, J. Large parallel computers. *J. ACM 13*, 1 (Jan. 1966), 25–32.

    Considers various classes of machines incorporating parallelism, outlines a general class of large-scale multiprocessors, and discusses the problems of hardware and software implementation.

Course A3. Analog and Hybrid Computing (2-2-3)

APPROACH

This course is concerned with analog, hybrid, and related digital techniques for solving systems of ordinary and partial differential equations, both linear and nonlinear. A portion of the course should be devoted to digital languages for the simulation of continuous or hybrid systems (MIDAS, PACTOLUS, DSL/90, etc.). The course will have both lecture and laboratory sessions. The laboratory will allow the students to solve some problems on analog and/or hybrid computers and other problems through digital simulation of analog or hybrid computers. (Digital simulators of analog computers are now available for digital machines of almost any size [13-22]. Some simulators are written in problem oriented languages such as Fortran and may be adapted to almost any computer.)

CONTENT

1. *Basic Analog Components.* Addition, multiplication by a constant, integration, function generation, multiplication, division, square roots, noise generation, and other operations. Laboratory assignments are used to familiarize the student with the operation of the components. (15%)

2. *Solution of Differential Equations.* The block-oriented approach to the solution of linear, nonlinear, and partial differential equations. Magnitude and time scaling. Estimation of maximum values. Equations with forcing functions and variable coefficients. Simultaneous equations. Statistical problems. (20%)

3. *Analog Computer Hardware.* Description of various analog computers. Amplifiers, potentiometers, and other linear and nonlinear components. The patch board and the control panel. Recording and display equipment. Slow and repetitive operation. Several laboratory assignments to give the student "hands-on" experience with the available machines. (15%)

4. *Hybrid Computer Systems.* Different types of hybrid systems. Patchable logic and mode control. Comparators, switches, and different types of analog memories. Control of initial conditions or parameters. (15%)

5. *Analog and Digital Conversion.* Brief treatment of analog-to-digital and digital-to-analog conversion. Methods of conversion, sampling, interpolation, smoothing. Accuracy and speed considerations. Multiplexing of analog-to-digital converters. (10%)

6. *Digital Simulation of Analog and Hybrid Systems.* Comparison of available languages. (One language should be presented in detail. The students should compare some of the previous analog solutions to those obtained by simulation.) (20%)

7. *Exams.* (5%)

ANNOTATED BIBLIOGRAPHY

In the citations which follow, applicable chapters are sometimes indicated in parentheses after the annotation.

General textbooks or references for the major part of the course

1. Ashley, J. R., *Introduction to Analog Computation.* Wiley, New York, 1963, 294 pp.

   A compact textbook which stresses the use rather than the design aspects of analog computing. The text could be used for an undergraduate course, but hybrid computers would have to be covered from separate sources. (All chapters)

2. Carlson, A., Hannauer, G., Carey, T., and Holsberg, P. In *Handbook of Analog Computation.* Electronics Associates, Inc., Princeton, N. J., 1965.

   Although this manual is oriented to EAI equipment, it is a good basic reference and has an extensive bibliography on selected applications.

3. Fifer, S. *Analog Computation, Volumes I–IV.* McGraw-Hill, New York, 1961, 1,331 pp. CR-6126-1122.

   This series of four volumes contains a complete coverage of analog computers, including hardware and applications. It is a good source of problems and references up to 1961.

4. Huskey, H. D., and Korn, G. A. (Eds.) *Computer Handbook.* McGraw-Hill, New York, 1962, 1,225 pp. CR-6234-2179.

   This comprehensive volume on both analog and digital computers is somewhat hardware-oriented although applications are also included.

5. Jackson, A. S. *Analog Computation.* McGraw-Hill, New York, 1960, 652 pp. CR-6015-0190.

   Although now somewhat out-of-date, this is an excellent text for serious students in engineering, especially those with an interest in feedback-control theory. The text has many references and an appendix with problem sets. (Chaps. 2–5, 7, 8, 11, 14)

6. JENNESS, R. R. *Analog Computation and Simulation: Laboratory Approach.* Allyn and Bacon, Boston, 1955, 298 pp.

Two parts: the first introduces the analog computer; the second gives the solutions of 21 problems in great detail.

7. JOHNSON, C. L. *Analog Computer Techniques,* 2nd ed. McGraw-Hill, New York, 1963, 336 pp. CR-6451-5162.

The text assumes a knowledge of basic electrical and mathematical principles. Some parts require an understanding of servo-mechanism theory and the Laplace transform. Each chapter has references and problems. (Chaps. 1–3, 7, 10, 12)

8. KARPLUS, W. J. *Analog Simulation, Solution of Field Problems.* McGraw-Hill, New York, 1958, 434 pp. CR-6123-0729.

A reference on analog techniques for partial differential equations. Includes material on the mathematical background for analog study of field problems, a description of analog hardware, and a mathematical discussion of applications of analog techniques to different classes of differential equations. Problem-oriented. Extensive bibliographies.

9. KARPLUS, W. J., AND SOROKA, W. J. *Analog Methods,* 2nd ed. McGraw-Hill, New York, 1959, 496 pp.

Three parts: on indirect computing elements; on indirect computers; and on direct computers. Describes both electromechanical and electronic computers and covers applications comprehensively.

10. KORN, G. A., AND KORN, T. M. *Electronic Analog and Hybrid Computers.* McGraw-Hill, New York, 1963, 584 pp. CR-6562-7468.

This text covers the theory, design, and application of analog and hybrid computers and has one of the most complete bibliographies available. Its compactness makes it more suitable for an undergraduate text.

11. LEVINE, L. *Methods for Solving Engineering Problems Using Analog Computers.* McGraw-Hill, New York, 1964, 485 pp. CR-6455-6477.

One of the best available texts, but it needs supplementing on equipment. (Ref. 2 might be good for this purpose.) In addition to the usual topics, there are chapters on optimization techniques, estimation and testing of hypotheses, and applications in statistics. (Chaps. 1–7).

12. SMITH, G. W., AND WOOD, R. C. *Principles of Analog Computation.* McGraw-Hill, New York, 1959, 234 pp. CR-6121-0411.

Introduces analog computers and illustrates various programming techniques in simulation and computation.

Descriptions of digital simulators of continuous systems

13. BRENNAN, R. D., AND SANO, H. PACTOLUS—a digital analog simulator program for the IBM 1620. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 299–312. CR-6563-7630.

Well-written article describes a digital program for the simulation of an analog computer. The program is written in FORTRAN and may be adapted to most machines. It allows man-machine interaction.

14. FARRIS, G. J., AND BURKHART, L. E., The DIAN digital simulation program. *Simulation 6,* 5 (May 1966), 298–304.

Describes a digital computer program which has some of the features of a digital differential analyzer and which is particularly suitable for the solution of boundary value problems.

15. HARNETT, R. T., AND SANSOM, F. J. *MIDAS Programming Guide.* Report No. SEG-TDR-64-1, Analog Comput. Divn., Syst. Engng. Group, Res. and Techn. Divn., US Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, Jan. 1964. CR-6672-9510.

This is the programming manual for the MIDAS simulation language. It is well-written and contains four examples complete with problem descriptions, block diagrams, coding sheets, and computed results.

16. JANOSKI, R. M., SCHAEFER, R. L., AND SKILES, J. J. COBLOC—a program for all-digital simulation of a hybrid computer. *IEEE Trans. EC-15,* 2 (Feb. 1966), 74–91.

COBLOC is a compiler which allows all-digital simulation of a hybrid computer having both analog and digital computation capability.

17. MORRIS, S. M., AND SCHIESSER, W. E. Undergraduate use of digital simulation. *Simulation 7,* 2 (Aug. 1966), 100–105. CR-6781-11,027.

Describes the LEANS (Lehigh Analog Simulator) program and shows the solution of a sample problem.

18. RIDEOUT, V. C., AND TAVERNINI, L. MADBLOC, a program for digital simulation of a hybrid computer. *Simulation 4,* 1 (Jan. 1965), 20–24.

Gives a brief description of the hybrid simulation language MADBLOC (one of the Wisconsin "BLOC" programs) which is written in the MAD language. Parameter optimization of a simple feedback system is given as an example.

19. STEIN, M. L., ROSE, J., AND PARKER, D. B. A compiler with an analog oriented input language. *Simulation 4,* 3 (Mar. 1965), 158–171.

Gives a description of a compiler program called "ASTRAL" which accepts analog oriented statements and produces FORTRAN statements. It is a reprint of the same paper from the Proc. of 1959 Western Joint Comput. Conf.

20. SYN, W. M., AND LINEBARGER, R. M. DSL/90—a digital simulation program for continuous system modeling. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 165–187. CR-6676-10,708.

A program which accepts block-oriented statements and compiles them into FORTRAN IV statements. Mixing of DSL/90 and FORTRAN statements is allowed. The program is available for the IBM 7090/94 and 7040/44.

Comparisons of digital simulators for continuous systems

21. LINEBARGER, R. N., AND BRENNAN, R. D. A survey of digital simulation: digital analog simulator programs. *Simulation 3,* 6 (Dec. 1964), 22–36. CR-6671-9009.

Gives a brief account of the following digital-analog simulation languages: SELFRIDE, DEPI, ASTRAL, DEPI4, DYSAC, PARTNER, DAS, JANIS, MIDAS, and PACTOLUS

22. LINEBARGER, R. N., AND BRENNAN, R. D. Digital simulation for control system design. *Instr. Contr. Syst. 38,* 10 (Oct. 1965), 147–152. CR-6675-10,425.

This paper has a very complete bibliography of digital simulators and a table classifying them.

Course A4. System Simulation (3-0-3)

APPROACH

This course can be taught from several different points of view: simulation can be treated as a tool of applied mathematics; it can be treated as a tool for optimization in operations research; or it can be treated as an example of the application of computer science techniques. Which orientation is used and the extent to which computer programs are an integral part of the course should depend upon the interests of the instructor and the students. Most instructors will find it useful to require several small programs and a term project. The availability of a simulation language for student use is desirable.

CONTENT

The numbers in square brackets refer to the items listed in the bibliography which follows.

1. *What is simulation?* (5%) [1, 2, 3]
   a. Statistical sampling experiment. [20]
   b. Comparison of simulation and other techniques. [7]
   c. Comparison of discrete, continuous, and hybrid simulation. [12, 21]
2. *Discrete Change Models.* (20%)
   a. Queueing models. [13]
   b. Simulation models. [1, 2, 3]

3. *Simulation languages.* (20%) [16, 21, 22]
4. *Simulation Methodology.* (20%) [1, 2, 3]
   a. Generation of random numbers and random variates. [14]
   b. Design of experiments and optimization. [6, 8, 9, 17]
   c. Analysis of data generated by simulation experiments. [10, 11, 15]
   d. Validation of models and results. [8, 19]
5. *Selected Applications of Simulation.* (10%) [4, 5]
   a. Business games.
   b. Operations research. [18]
   c. Artificial intelligence.
6. *Research Problems in Simulation Methodology.* (5%)
7. *Term Project.* (20%)

BIBLIOGRAPHY

Textbooks covering a number of the topics of this course

1. Chorafas, D. N. *Systems and Simulation.* Academic Press, New York, 1965, 503 pp.
2. Naylor, T. H., Balintfy, J. L., Burdick, D. S., and Chu, K. *Computer Simulation Techniques.* Wiley, New York, 1966, 352 pp. CR-6781-11,103.
3. Tocher, K. D. *The Art of Simulation.* D. Van Nostrand, Princeton, N. J., 1963, 184 pp. CR-6454-6091.

Bibliographies devoted to simulation

4. IBM Corporation. *Bibliography on Simulation.* Report 320-0924-0, 1966.
5. Shubik, M. Bibliography on simulation, gaming, artificial intelligence, and allied topics. *J. Amer. Statist. Assoc. 55,* 292 (Dec. 1960), 736–751. CR-6122-0581.

Other works on simulation (which also contain extensive bibliographies)

6. Burdick, D. S., and Naylor, T. Design of computer simulation experiments for industrial systems. *Comm. ACM 9,* 5 (May 1966), 329–338. CR-6783-11,714.
7. Connors, M. M., and Teichroew, D. *Optimal Control of Dynamic Operations Research Models.* International Textbook Co., Scranton, Pa., 1967, 118 pp.
8. Conway, R. W. Some tactical problems in digital simulation. *Management 10,* 1 (Oct. 1963), 47–61.
9. Ehrenfeld, S., and Ben-Tuvia, S. The efficiency of statistical simulation procedures. *Technometrics 4,* 2 (May 1962), 257–276. CR-6341-3742.
10. Fishman, G. S. Problems in the statistical analysis of simulation experiments: The comparison of means and the length of sample records. *Comm. ACM 10,* 2 (Feb. 1967), 94–99. CR-6783-12,103.
11. Fishman, G. S., and Kiviat, P. J. The analysis of simulation-generated time series. *Mgmt. Sci. 13,* 7 (Mar. 1967), 525–557.
12. Forrester, J. W. *Industrial Dynamics.* M.I.T. Press, Cambridge, Mass., and Wiley, New York, 1961, 464 pp.
13. Galliher, H. Simulation of random processes. In *Notes on Operations Research,* Operations Research Center, M.I.T., Cambridge, Mass., 1959, pp. 231–250.
14. Hull, T. E., and Dobell, A. R. Random number generators. *SIAM Rev. 4,* 3 (July 1962), 230–254. CR-6341-3749.
15. Jacoby, J. E., and Harrison, S. Multivariable experimentation and simulation models. *Naval Res. Log. Quart. 9,* (1962), 121–136.
16. Krasnow, H. S., and Merikallio, R. The past, present and future of general simulation languages. *Mgmt. Sci. 11,* 2 (Nov. 1964), 236–267. CR-6566-8521.
17. McArthur, D. S. Strategy in research—alternative methods for design of experiments. *IRE Trans. Eng. Man. EM-8,* 1 (Jan. 1961), 34–40.
18. Morgenthaler, G. W. The theory and application of simulation in operations research. In Russel L. Ackoff (Ed.), *Progress in Operations Research,* Wiley, New York, 1961, pp. 363–419.
19. Schenk, H., Jr. Computing "AD ABSURDUM." *The Nation 196,* 12 (June 15, 1963), 505–507.
20. Teichroew, D. A history of distribution sampling prior to the era of the computer and its relevance to simulation. *J. Amer. Statist. Assoc. 60,* 309 (Mar. 1965), 27–49. CR-6673-9823.
21. Teichroew, D. Computer simulation—discussion of the technique and comparison of languages. *Comm. ACM 9,* 10 (Oct. 1966), 723–741. CR-6782-11,466.
22. Tocher, K. D. Review of simulation languages. *Oper. Res. Quart. 15,* 2 (June 1965), 189–218.

Course A5. Information Organization and Retrieval (3-0-3)

APPROACH

This course is designed to introduce the student to information organization and retrieval of natural language data. Emphasis should be given to the development of computer techniques rather than philosophical discussions of the nature of information. The applicability of the techniques developed for both data and document systems should be stressed. The student should become familiar not only with the techniques of statistical, syntactic and logical analysis of natural language for retrieval, but also with the extent of success or failure of these techniques. The manner in which the techniques may be combined into a system for use in an operational environment should be explored. In the event that a computer is available together with natural language text in a computer-readable form, programming exercises applying some of the techniques should be assigned. If this is not possible, the student should present a critique and in-depth analysis of an article selected by the instructor.

CONTENT

1. *Information Structures.* Graph theory, document and term-document graphs, semantic road maps, trees and lists, thesaurus and hierarchy construction, and multilists.
2. *Dictionary Systems.* Thesaurus look-up, hierarchy expansion, and phrase dictionaries.
3. *Statistical Systems.* Frequency counts, term and document associations, clustering procedures, and automatic classification.
4. *Syntactic Systems.* Language structure, automatic syntactic analysis, graph matching, and automatic tree matching.
5. *Vector Matching and Search Strategies.* Keyword matching, direct and inverted files, combined file systems, correlation functions, vector merging and matching, matching of cluster vectors, and user feedback systems.
6. *Input Specifications and System Organization.* Input options. Supervisory systems, their general organization, and operating procedures.
7. *Output Systems.* Citation indexing and bibliographic coupling. Secondary outputs including concordances, abstracts, and indexes. Selective dissemination. Catalog systems.
8. *Evaluation.* Evaluation environment, recall and precision, presentation of results, and output comparison.
9. *Automatic Question Answering.* Structure and extension of data bases, deductive systems, and construction of answer statements.

ANNOTATED BIBLIOGRAPHY

There is no one book currently available that could be used as a text for this course, so most of the material must be obtained from the literature. References 2 and 6 provide excellent state-of-the-art surveys and guides to the literature.

1. Becker, J., and Hayes, R. M. *Introduction to Information Storage and Retrieval: Tools, Elements, Theories.* Wiley, New York, 1963, 448 pp.
   A standard textbook, perhaps the best of those presently available.

2. Cuadra, C. (Ed.) *Annual Review of Information Science and Technology.* Interscience, New York, Vol. 1, 1966 and Vol. 2, 1967.

A survey and review publication.

3. Hays, D. G. (Ed.) *Readings in Automatic Language Processing.* American Elsevier, New York, 1966.

Includes examples of text processing applications.

4. Salton, G. Progress in automatic information retrieval. *IEEE Spectrum 2*, 8 (Aug. 1965), 90–103. CR-6675-10,409.

A survey of current capabilities in text processing.

5. Salton, G. *Automatic Information Organization and Retrieval.* McGraw-Hill, New York, to be published in 1968.

A text concentrating on automatic computer-based information retrieval systems.

6. Stevens, M. E. *Automatic Indexing: A State-of-the-Art Report.* Monograph 91, National Bureau of Standards, US Dept. of Commerce, Washington, D.C., March 30, 1965.

A survey article that covers the historical development of automatic indexing systems through 1964.

7. Stevens, M. E., Giuliano, V. E., and Heilprin, L. B. (Eds.) *Statistical Association Methods for Mechanized Documentation —Symposium Proceedings.* Miscellaneous Publication 269, US Dept. of Commerce, Washington, D. C., 1964.

A collection of papers concerned with statistical association techniques.

Course A6. Computer Graphics (2-2-3)

APPROACH

Since this field is basically only a few years old, it is not surprising that no underlying theories are uniformly accepted by the researchers and implementers. Rather, the pertinent information exists as a number of loosely related project descriptions in conference proceedings and professional journals. This situation is similar to that in the information retrieval field, where those conducting courses at a number of major universities report on current accomplishments and research in an effort to coordinate and structure the mass of available information and to teach those techniques which have been found useful. Thus, for the present, this course probably should be taught as a seminar where the literature is read and perhaps reported by selected students. After some texts become available and after more experience has been gained, a more formal course atmosphere could be established.

Although the literature is plentiful, this course clearly assumes substantial value to the student only when it includes an intensive laboratory (hopefully using a display console) where the various algorithms can be tested, compared, and extended. The laboratory periods are meant to be used for explaining the details of algorithms or hardware and software that are not appropriate to a more formal lecture. A variety of programming projects in pattern recognition or display programming, for example, are within the scope of a one-semester course. The time spent on these projects would be in addition to the laboratory time.

CONTENT

The thread running through the topics listed below is the unit of information—the picture. The course should deal with common ways in which the picture is handled in a variety of hitherto largely unrelated disciplines. First hardware and then software topics should be considered, since the software today is still a function of present hardware.

The order and depth of coverage of the material suggested below is quite flexible—another sensible order of the material might be topics 1, 2, 6, 7, 8, and 3, 4, 5 optional, which would then constitute a course in displays. (In any case some material in Sections 6 and 7 would have to be covered briefly to prepare students for their projects.) Also an entire semester could be devoted to pattern recognition.

1. Motivation for graphical data processing and its history, particularly that of displays. (5%)

2. Brief introduction to psycho-physical photometry and display-parameters such as resolution and brightness. Block diagram of display systems and delineation of the functions of their components: the computer subsystem; buffer or shared memory; command decoder; display generator; and producer(s) of points, lines, vectors, conic sections, and characters. Extended capabilities such as subroutining, windowing, hardware matrix operations, and buffer manipulation. Comparison of various types of CRTs with other display producing techniques such as photochromics and electroluminescence. Brief discussion of passive graphics (output only) devices such as x-y plotters, and microfilm recorders. Interrupts, manual inputs and human interaction with active displays via light pens, voltage pens, function keys, tablets, wands, joy sticks, etc. Demonstration of equipment. (10%)

3. Contrast of information retrieval with document retrieval and definition of indexing and locating. Image recording parameters such as resolution, and their comparison with display parameters. Demonstration or discussion of microfilm and microfilm handling devices (manual and automated). Brief discussion of photochromics, thermoplastics and other nonconventional media. Brief discussion of electro-optical techniques for recording, modulating, and deflecting. (5%)

4. Scanning and digitizing of paper or film, and subsequent transmission of digitized information, including band-width/cost tradeoffs. Brief review of digital storage techniques and parameters, and discussion of tradeoffs in bulk digital versus image storage for pictorial and digital data. Introduction of the notion of a combination of a digital and an image system. (5%)

5. Digitizing as an input process for pattern description and recognition and preprocessing of this input (cosmetology and normalization). Contrast of symbol manipulative, linguistic, and mathematical techniques, such as gestalt, caricatures, features, moments, random nets, decision functions, syntax-directed techniques and real-time tracking techniques using scopes and tablets. Electro-optical techniques such as optical Fourier transforms may also be covered briefly. (20%)

6. Picture models and data structures. Geometry, topology, syntax, and semantics of pictures, stressing picture/subpicture hierarchy. The differences between block diagram, wire frame, and surface representations. Use of tables, trees, lists, plexes, rings, associative memory, and hashing schemes for data structures, and the data structure (or list processing) languages which create and manipulate them. Mathematics of constraint satisfaction, windowing, three-dimensional transformations and projections, and hidden-line problems. (25%)

7. Display software (probably specific to a given installation). Creation and maintenance of the display file, translations between the data structure and the display file, interrupt handling, pen pointing and tracking, and correlation between light pen detects and the data structure. Use of macros or compiler level software for standard functions. Software for multiconsole time-shared graphics with real-time interaction. (20%)

8. Selected applications: (10%)
   a. Menu programming and debugging
   b. Flowchart and block diagram processing
   c. Computer assisted instruction
   d. Computer aided design
   e. Chemical modeling
   f. Business
   g. Animated movies

ANNOTATED BIBLIOGRAPHY

This bibliography is not complete in its coverage and consists primarily of survey articles, as no textbooks exist. A number of the more detailed technical articles in the literature were omitted because they were concerned with specific situations or machines. Most of the survey articles listed have good bibliographies. A selection of topics from the outline above can thus be followed by a selection of appropriate research papers from the literature.

1. Fetter, W. A. *Computer Graphics in Communications.* McGraw-Hill, New York, 1965, 110 pp.

   This volume is strong on engineering applications and illustrations.

2. Gray, J. C. Compound data structures for computer-aided design: a survey. Proc. ACM 22nd Nat. Conf., 1967, Thompson Book Co., Washington, D. C., pp. 355–366.

   A brief survey and comparison of various types of data structures currently in use.

3. Gruenberger, F. (Ed.) *Computer Graphics: Utility/Production/Art.* Thompson Book Co., Washington, D. C., 1967, 225 pp.

   Collection of survey papers, useful for orientation.

4. Narasimhan, R. Syntax-directed interpretation of classes of pictures. *Comm. ACM 9*, 3 (Mar. 1966), 166–173.

   An introduction to syntactic descriptive models for pictures, implemented using simulated parallel processing. This linguistic approach is also taken by Kirsch, Grenander, Miller, and others.

5. Parker, D. B. Solving design problems in graphical dialogue. In W. J. Karplus (Ed.), *On-Line Computing*, McGraw-Hill, 1967, pp. 176–219.

   A software-oriented survey of display console features.

6. Ross, D. G., and Rodriguez, J. E. Theoretical foundations for the computer-aided design system. Proc. AFIPS 1963 Spring Joint Comput. Conf., Vol. 23, Spartan Books, New York, pp. 305–322.

   Introduction of the "plex" as a compound list structure for both graphical and nongraphical entities. Outline of the AED philosophy and algorithmic theory of language.

7. Sutherland, I. E. Sketchpad: a man-machine graphical communication system. Lincoln Lab. Tech. Rep. No. 296, M.I.T., Lexington, Mass., 1963, 91 pp.

   Presents the pace-setting Sketchpad system: its capabilities, data structure, and some implementation details.

8. Sutherland, W. R. The on-line graphical specification of computer procedures. Ph.D. Dissertation, M.I.T., Cambridge, Mass., Jan. 1966, and Lincoln Lab. Tech. Rep. No. 405, May 1966.

   Describes a graphical language, executed interpretively, which avoids written labels and symbols by using data connections between procedure elements to determine both program flow and data flow.

9. van Dam, A. Bibliography on computer graphics. *ACM SIGGRAPHICS Newsletter 1*, 1 (Apr. 1967), Association for Computing Machinery, New York.

   This extensive bibliography is being kept up-to-date in successive issues of the *Newsletter*.

10. van Dam, A. Computer-driven displays and their uses in man/machine interaction. In F. L. Alt (Ed.), *Advances in Computers, Vol. 7*, Academic Press, New York, 1966, pp. 239–290.

    A hardware-oriented description of CRT console functions.

Course A7. Theory of Computability (3-0-3)

APPROACH

This is a theoretical course which should be taught in a formal and precise manner, i.e. definitions, theorems, and proofs. The theory of recursive functions and computability should, however, be carefully motivated and illustrated with appropriate examples.

CONTENT

More material is listed than can easily be covered in a three-hour one-semester course. The first three topics should definitely be covered, but the instructor can select material from the remaining topics.

1. Introduction to Turing machines (TM's) and the invariance of general computability under alterations of the model. Wang machines, Shepherdson-Sturgis machines, machines with only 2 symbols, machines with only 2 states, machines with nonerasing tapes, machines with multiple heads and multidimensional tapes. (4 lectures)

2. Universal Turing machines. (2 lectures)

3. Gödel numbering and unsolvability results, the halting problem, and Post's correspondence problem. (3 lectures)

4. Relative uncomputability, many-one reducibility and Turing reducibility, and the Friedberg-Muchnik theorem. (6 lectures)

5. TM's with restricted memory access, machines with one counter, pushdown automata and their relation to context-free languages. Universality of machines with two counters. (3 lectures)

6. TM's with limited memory, linear bounded automata and their relation to context-sensitive languages, and the Stearns-Hartmanis-Lewis hierarchy. (5 lectures)

7. TM's with limited computing time and the Hartmanis-Stearns time hierarchy. (5 lectures)

8. Models for real-time computation, TM's with many tapes versus 1 or 2 tapes, and TM's with many heads per tape versus 1 head per tape. (4 lectures)

9. Random-access stored-program machines, iterative arrays, general bounded activity machines, n-counter real-time machines, and other computing devices. (8 lectures)

10. Complexity classification by functional definition, primitive recursive functions, the Grzegorczyk hierarchy and its relation to Algol programming, real-time countability, and an algorithm for fast multiplication. (6 lectures)

ANNOTATED BIBLIOGRAPHY

1. Aanderaa, S., and Fischer, P. C. The solvability of the halting problem for 2-state Post machines. *J. ACM 14*, 4 (Oct. 1967), 677–682.

   A problem unsolvable for quintuple Turing machines is shown to be solvable for the popular quadruple version of Post.

2. Caianiello, E. R. (Ed.) *Automata Theory.* Academic Press, New York, 1966, 342 pp. CR-6676-10,935.

   A collection of research and tutorial papers on automata, formal languages, graph theory, logic, algorithms, recursive function theory, and neural nets. Because of varying interest and difficulty, the papers might be useful for supplementary reading by ambitious students.

3. Davis, M. *Computability and Unsolvability.* McGraw-Hill, New York, 1958, 210 pp.

   Contains an introduction to the theory of recursive functions, most of Kleene's and Post's contributions to the field and some more recent work.

4. Davis, M. (Ed.) *The Undecidable—Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions.* Raven Press, Hewlett, New York, 1965, 440 pp. CR-6673-9790.

   An anthology of the fundamental papers of Church, Gödel, Kleene, Post, Rosser, and Turing on undecidability and unsolvability.

5. Fischer, P. C. Multitape and infinite-state automata—a survey. *Comm. ACM 8*, 12 (Dec. 1965), 799–805. CR-6675-10,561.

   A survey of machines which are more powerful than finite automata and less powerful than Turing machines. Extensive bibliography.

6. Fischer, P. C. On formalisms for Turing machines. *J. ACM 12*, 4 (Oct. 1965), 570–580. CR-6675-10,558.

Variants of one-tape Turing machines are compared and transformations from one formalism to another are analyzed.

7. Friedberg, R. M. Two recursively enumerable sets of incomparable degrees of unsolvability (Solution of Post's Problem, 1944). *Proc. Nat. Acad. Sci. 43*, (1957), 236–238.
   The "priority" method for generating recursively enumerable sets is introduced and used to solve this famous problem.

8. Ginsburg, S. *Mathematical Theory of Context-Free Languages*. McGraw-Hill, New York, 1966, 243 pp.
   The first textbook on the theory of context-free languages. It gives a detailed mathematical treatment of pushdown automata, ambiguity, and solvability.

9. Hartmanis, J., and Stearns, R. E. On the computational complexity of algorithms. *Trans. AMS 117*, 5 (May 1965), 285–306.
   Turing computable sequences are classified in terms of the rate with which a multitape Turing machine can output the terms of the sequence, i.e. the "Hartmanis-Stearns time hierarchy."

10. Hermes, H. *Enumerability, Decidability, Computability*. Academic Press, New York, 1965, 245 pp. CR-6673-9781.
    A systematic introduction to the theory of recursive functions, using Turing machines as a base.

11. Kleene, S. C. *Mathematical Logic*. Wiley, New York, 1967, 398 pp.
    A thorough yet elementary treatment of first-order mathematical logic for undergraduates. Contains much of the material of the author's graduate text, *Introduction to Metamathematics*, (D. Van Nostrand, Princeton, N. J., 1952, 550 pp.). The material has been updated and reorganized to be more suitable for the beginning student.

12. McNaughton, R. The theory of automata, a survey. In F. L. Alt (Ed.), *Advances in Computers, Vol. 2*. Academic Press, New York, 1961, pp. 379–421. CR-6342-3920.
    Most of the areas of automata theory are included with the exception of switching theory and other engineering topics. A list of 119 references.

13. Minsky, M. L. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N. J., 1967, 317 pp.
    The concept of an "effective procedure" is developed. Also treats algorithms, Post productions, regular expressions, computability, infinite and finite-state models of digital computers, and computer languages.

14. Myhill, J. Linear bounded automata. *WADD Tech. Note 60-165*, Wright-Patterson Air Force Base, Ohio, 1960.
    The paper which first defined a new class of automata whose power lies between those of finite automata and Turing machines.

15. Post, E. L. Recursive unsolvability of a problem of Thue. *J. Symbol. Logic 11*, (1947), 1–11.
    Contains results on one variant of the "word problem" for semigroups using Turing machine methods.

16. Rogers, H., Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1967, 482 pp.
    A current and comprehensive account of recursive function theory. Proceeds in an intuitive semiformal manner, beginning with the recursively enumerable sets and ending with the analytical hierarchy.

17. Shannon, C. E., and McCarthy, J. (Eds.) *Automata Studies*. Princeton University Press, Princeton, N. J., 1956, 285 pp. CR-6565-8330.
    A collection of many of the early papers on finite automata, Turing machines, and synthesis of automata which stimulated the development of automata theory. Philosophical papers, in addition to mathematical papers, are included since the aim of the collection is to help explain the workings of the human mind.

18. Shepherdson, J. C., and Sturgis, H. E. Computability of recursive functions. *J. ACM 10*, 2 (Apr. 1963), 217–255. CR-6451-5105.
    A class of machines which is adequate to compute all partial recursive functions is obtained by relaxing the definition of a Turing machine. Such machines can be easily designed to carry out some specific intuitively effective procedure.

19. Stearns, R. E., Hartmanis, J., and Lewis, P. M. Hierarchies of memory limited computations. *1965 IEEE Conference Record on Switching Circuit Theory and Logic Design*, Special Publication 16 C 13, Institute of Electrical and Electronic Engineers, New York, Oct. 1965, pp. 179–190.
    Turing computable functions are classified according to the relationship of the amount of storage required for a computation to the length of the input to the computation, i.e. the "Stearns-Hartmanis-Lewis hierarchy."

20. Trakhtenbrot, B. A. *Algorithms and Automatic Computing Machines*, transl. by J. Kristian, J. D. McCawley, and S. A. Schmitt. D. C. Heath, Boston, 1963, 101 pp.
    A translation and adaptation from the second Russian edition (1960) of the author's elementary booklet on solvability and Turing machines.

21. Turing, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc., Ser. 2, 42*, (1936–1937), pp. 230–265.
    The famous memoir on decision problems which initiated the theory of automata.

22. Von Neumann, J. *Theory of Self-Reproducing Automata*. (Edited and completed by A. W. Burks.) University of Illinois Press, Urbana, Illinois, 1966, 388 pp. CR-6700-0670.
    Consists of all previously unpublished sections of Von Neumann's general theory of automata. Part I includes the kinematic model of self-reproduction. Part II, which is much longer, treats the logical design of a self-reproducing cellular automaton.

23. Wang, H. A variant to Turing's theory of computing machines. *J. ACM 4*, 1 (Jan. 1957), 61–92.
    An abstract machine is defined which is capable of carrying out any computation and which uses only four basic types of instructions in its programs.

Course A8. Large-scale Information Processing Systems (3-0-3)

APPROACH

This course is intended to give the student some appreciation of how computers fit into information systems and how information systems fit into a "large organization framework." As this field is evolving rapidly, the most interesting and relevant material appears in articles; moreover, the field is so large that not all the relevant material can be covered. The course may be conducted as a lecture course, but assignment of individual readings in a seminar-type situation might be more suitable.

Many information processing systems are so large that they require a number of computer programs to be run on a continuing basis using large quantities of stored data. The process of establishing such a large system involves a number of steps: (1) the determination of the processing requirements; (2) the statement of those requirements in a complete and unambiguous form suitable for the next steps; (3) the design of the system, i.e. the specification of computer programs, hardware devices, and procedures which together can "best" accomplish the required processing; (4) the construction of the programs and procedures, and the acquisition of the hardware devices; and (5) the testing and operation of the assembled components in an integrated system. This course is designed to help prepare the student to participate in the development of such systems.

CONTENT

The numbers in square brackets after each topic listed below refer to the items listed in the bibliography which follows.

1. Examples of large-scale information systems. (10%) [12, 28, 30]

a. Computer centers. [20, 27, 34]
b. Information retrieval. [2]
c. Real-time and time-sharing. [15, 16, 33]
d. Business data processing. [14, 18, 19, 31, 32]

2. Data structures and file management systems. (20%) [1, 4, 5, 9, 10, 13, 17, 29, 38]
3. Systems design methodology. (40%) [22, 32]
a. "Nonprocedural" languages. [8, 26, 38, 40]
b. Systems design. [3, 11, 21, 23, 24, 25, 36, 37]
c. Evaluation. [7]
4. Implementation problems. (10%) [6, 35]
5. Term project. (20%)

BIBLIOGRAPHY

1. Baum, C., and Gorsuch, L. (Eds.) Proceedings of the second symposium on computer-centered data base systems. TM-2624/100/00, System Development Corporation, Santa Monica, Calif., 1 Dec. 1965.
2. Berul, L. Information storage and retrieval, a state-of-the-art report. Report AD-630-089, Auerbach Corporation, Philadelphia, Pa., 14 Sept. 1964.
3. Briggs, R. A mathematical model for the design of information management systems. M.S. Thesis, U. of Pittsburgh, Pittsburgh, Pa., 1966.
4. Brooks, F. P., Jr., and Iverson, K. E. *Automatic Data Processing.* Wiley, New York, 1963, 494 pp.
5. Bryant, J. H., and Semple, P., Jr. GIS and file management. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 97-107.
6. Buchholz, W. (Ed.) *Planning a Computer System.* McGraw-Hill, New York, 1962, 322 pp. CR-6346-4786.
7. Calingaert, P. System evaluation: survey and appraisal. *Comm. ACM 10,* 1 (Jan. 1967), 12-18. CR-6782-11,661.
8. Codasyl Development Committee, Language Structure Group. An information algebra, phase I report. *Comm. ACM 5,* 4 (Apr. 1962), 190-201. CR-6235-2621.
9. Connors, T. L. ADAM—generalized data management system. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 193-203. CR-6676-10,822.
10. Control Data Corporation. *3600/3800 INFOL Reference Manual.* Publ. No. 60170300, CDC, Palo Alto, Calif., July, 1966.
11. Day, R. H. On optimal extracting from a multiple file data storage system: an application of integer programming. *J. ORSA 13,* 3 (May-June, 1965), 482-494.
12. Desmonde, W. H. *Computers and Their Uses.* Prentice-Hall, Englewood Cliffs, N. J., 1964, 296 pp. CR-6561-6829.
13. Dobbs, G. H. State-of-the-art survey of data base systems. Proc. Second Symposium on Computer-Centered Data Base Systems, TM-2624/100/00, System Development Corporation, Santa Monica, Calif., 1 Dec. 1965, pp. 2-3 to 2-10.
14. Elliott, C. O., and Wasley, R. S. *Business Information Processing Systems.* Richard D. Irwin, Homewood, Ill., 1965, 554 pp.
15. Fife, D. W. An optimization model for time-sharing. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 97-104. CR-6676-10,869.
16. Franks, E. W. A data management system for time-shared file processing using a cross-index file and self-defining entries. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 79-86. CR-6676-10,754.
17. General Electric Company. *Integrated Data Store—A New Concept in Data Management.* Application Manual AS-CPB-483A, Revision of 7-67, GE Computer Division, Phoenix, Ariz., 1967.
18. Gotlieb, C. C. General purpose programming for business applications. In F. L. Alt (Ed.), *Advances in Computers, Vol. 1,* Academic Press, New York, 1960, pp. 1-42. CR-6016-0206.
19. Gregory, R. H., and Van Horn, R. L. *Automatic Data Processing Systems,* 2nd ed. Wadsworth Pub. Co., San Francisco, 1963, 816 pp. CR-6016-0301, of 1st ed.
20. Hutchinson, G. K. A computer center simulation project. *Comm. ACM 8,* 9 (Sept. 1965), 559-568. CR-6673-9617.
21. Katz, J. H. Simulation of a multiprocessor computer system. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 127-139. CR-6676-10,870.
22. Laden, H. N., and Gildersleeve, T. R. *System Design for Computer Application.* Wiley, New York, 1963, 330 pp.
23. Langefors, B. Some approaches to the theory of information systems. *BIT 3,* 4 (1963), 229-254. CR-6455-6399.
24. Langefors, B. Information system design computations using generalized matrix algebra. *BIT 5,* 2 (1965), 96-121.
25. Lombardi, L. Theory of files. Proc. 1960 Eastern Joint Comput. Conf., Vol. 18, Spartan Books, New York, pp. 137-141. CR-6236-3165.
26. Lombardi, L. A general business-oriented language based on decision expressions. *Comm. ACM 7,* 2 (Feb. 1964), 104-111. CR-6671-9013.
27. Lynch, W. C. Description of a high capacity, fast turnaround university computer center. *Comm. ACM 9,* 2 (Feb. 1966), 117-123. CR-6673-9546.
28. Maley, G. A., and Skiko, E. J. *Modern Digital Computers.* Prentice-Hall, Englewood Cliffs, N. J., 1964, 216 pp. CR-6561-7081.
29. McCabe, J. On serial files with relocatable records. *J. ORSA 13,* 4 (July-Aug. 1965), 609-618.
30. McCarthy, E. J., McCarthy, J., and Humes, D. *Integrated Data Processing Systems.* Wiley, New York, 1966, 565 pp.
31. McCracken, D. D., Weiss, H., and Lee, T.-H. *Programming Business Computers.* Wiley, New York, 1959, 510 pp. CR-6013-0076.
32. McGee, W. C. The formulation of data processing problems for computers. In F. L. Alt (Ed.) *Advances in Computers, Vol. 4,* Academic Press, New York, 1964, pp. 1-52.
33. Nielson, N. R. The simulation of time sharing systems. *Comm. ACM 10,* 7 (July 1967), 397-412.
34. Rosin, R. F. Determining a computer center environment. *Comm. ACM 8,* 7 (July 1965), 463-488.
35. Schultz, G. P., and Whister, T. L. (Eds.) *Management Organization and the Computer.* Free Press, Macmillan, New York, 1960, 310 pp.
36. Smith, J. L. An analysis of time-sharing computer systems using Markov models. Proc. AFIPS 1966 Spring Joint Comput. Conf., Vol. 28, Spartan Books, New York, pp. 87-95. CR-6676-10,835.
37. Turnburke, V. P., Jr. Sequential data processing design. *IBM Syst. J. 2,* (Mar. 1963), 37-48.
38. Ver Hoef, E. W. Design of a multilevel file management system. Proc. ACM 21st Nat. Conf., 1966, Thompson Book Co., Washington, D. C., pp. 75-86. CR-6781-11,185.
39. Young, J. W., Jr. Nonprocedural languages—a tutorial. Paper 7th Ann. Tech. Symposium, Mar. 23, 1965. South Calif. Chapters of ACM. Copies may be obtained from the author, Electronics Division, MS 50, National Cash Register, 2815 W. El Segundo Blvd., Hawthorne, Calif. 90750.
40. Young, J. W., Jr., and Kent, H. Abstract formulation of data processing problems. *J. Ind. Eng. 9,* 6 (Nov.-Dec. 1958), 471-479. (Also reprinted in *Ideas for Management, 1959.*)

Course A9. Artificial Intelligence and Heuristic Programming (3-0-3)

APPROACH

As this course is essentially descriptive, it might well be taught by surveying various cases of accomplishment in the areas under study. Each student should undertake some independent activity as part of his course work. This might take the form of a survey article on some aspect of the field: a program which simulates some of the rudimentary features of learning and forgetting; a program which plays some simple game like three-dimensional tic-tac-toe; or some other comparable activity. It would probably be best for the student to write any such programs in a list processing language.

CONTENT

The following outline is only a guide. Depending on the instructor's preferences and experience, variations will be introduced and new material will be added to the subject matter to be presented.

1. Definition of heuristic versus algorithmic methods using an example such as game playing. Description of cognitive processes taking place in deriving a new mathematical theorem. Outline of Polya's and Hadamard's approaches to mathematical invention. Discussion of the heuristic method as an exploratory and as an exclusive philosophy (cf. theorem proving à la Newell-Shaw-Simon, Robinson and Wang). Objectives, goals and purposes of work in areas under discussion. (3 lectures)

2. Game playing programs (chess, checkers, go, go-moku, bridge, poker, etc.). (3 lectures)

3. Theorem proving in logic and geometry. (3 lectures)

4. Formula manipulation on computers. (3 lectures)

5. Pattern recognition and picture processing. (3 lectures)

6. General problem solvers and advice takers. (4 lectures)

7. Question answering programs. (3 lectures)

8. Verbal and concept learning simulators. (3 lectures)

9. Decision making programs. (3 lectures)

10. Music composition by computers. (3 lectures)

11. Learning in random and structured nets. Neural networks. (3 lectures)

12. Adaptive systems. (3 lectures)

13. State-of-the-art in machine translation of languages and natural language processing. (4 lectures)

14. Questions of philosophical import: the mind-brain problem and the nature of intelligence, the relevance of operational definitions, and what is missing in present day "thinking machines." (2 lectures)

BIBLIOGRAPHY

The entries given below are grouped according to the items of the "Content" above to which they apply. This list serves only as a starting point and can be extended easily using the bibliographies listed below.

General reference

1. Feigenbaum, E. A., and Feldman, J. (Eds.) *Computers and Thought*. McGraw-Hill, New York, 1966, 535 pp. CR-6563-7473.
   Contains many of the articles listed below and a "Selected Descriptor-Indexed Bibliography" by Marvin Minsky.

Heuristic versus algorithmic methods [Item 1]

2. Armer, P. Attitudes toward intelligent machines. In *Computers and Thought*, pp. 389–405. CR-6125-0977 and CR-6236-2900.
3. Findler, N. V. Some further thoughts on the controversy of thinking machines. *Cybernetica 6*, (1963), 47–52.
4. Hadamard, J. *The psychology of invention in the mathematical field.* Dover Publications, New York, 1945, 145 pp. CR-6345-4614.
5. Minsky, M. Steps toward artificial intelligence. In *Computers and Thought*, pp. 406–450. CR-6232-1528.
6. Nagel, E. *The Structure of Science: Problems in the Logic of Scientific Explanation.* Harcourt, Brace & World, New York, 1961, 612 pp.
7. Polya, G. *Mathematics and Plausible Reasoning: Vol. I, Induction and Analogy in Mathematics; Vol. II, Patterns of Plausible Inference.* Princeton University Press, Princeton, N. J., 1954, 280 and 190 pp.

Game playing programs [Item 2]

8. Berlekamp, E. R. Program for double-dummy bridge problems—a new strategy for mechanical game playing. *J. ACM 10*, 3 (July 1963), 357–364. CR-6452-5297.
9. Findler, N. V. Computer models in the learning process. In *Proc. Internat. Symposium on Mathematical and Computational Methods in the Social and Life Sciences, Rome, 1966.*
10. Newell, A., Shaw, J. C., and Simon, H. A. Chess playing programs and the problem of complexity. In *Computers and Thought*, pp. 39–70. CR-6012-0048.
11. Pervin, I. A. On algorithms and programming for playing at dominoes, transl. from Russian. *Automation Express 1* (1959), 26–28. CR-6235-2328.
12. Remus, H. Simulation of a learning machine for playing Go. Proc. IFIP Congress, Munich, 1962, North-Holland Pub. Co., Amsterdam, pp. 192–194. CR-6341-3420.
13. Samuel, A. L. Some studies in machine learning using the game of checkers. In *Computers and Thought*, pp. 71–105.

Theorem proving in logic and geometry [Item 3]

14. Davis, M., Logemann, G., and Loveland, D. A machine program for theorem-proving. *Comm. ACM 5*, 7 (July 1962), 394–397.
15. Gelernter, H., Hansen, J. R., and Loveland, D. W. Empirical exploration of the geometry-theorem proving machine. In *Computers and Thought*, pp. 134–152. CR-6233-1928.
16. Newell, A., Shaw, J. C., and Simon, H. A. Empirical explorations with the logic theory machine: a case study in heuristics. In *Computers and Thought*, pp. 109–133.
17. Robinson, J. A. Theorem proving on the computer, *J. ACM 10*, 2 (Apr. 1963), 163–174. CR-6452-5460.
18. Wang, H. Proving theorems by pattern recognition. *Comm. ACM 3*, 4 (Apr. 1960), 220–234. CR-6016-0369.

Formula manipulation on computers [Item 4]

19. Bond, E., Auslander, M., Grisoff, S., Kenney, R., Myszewski, M., Sammet, J. E., Tobey, R. G., and Zilles, S. FORMAC—an experimental FORmula MAnipulation Compiler. Proc. ACM 19th Nat. Conf., 1964, Association for Computing Machinery, New York, pp. K2.1-1 to K2.1-19.
20. Brown, W. S. The ALPAK system for nonnumerical algebra on a digital computer, I and II. *Bell Syst. Tech. J. 42* (1963), 2081–2119, and *43* (1964), 785–804.
21. Perlis, A. J., and Iturriaga, R. An extension to ALGOL for manipulating formulae. *Comm. ACM 7*, 2 (Feb. 1964), 127–130.
22. Sammet, J. E. An annotated descriptor based bibliography on the use of computers for nonnumerical mathematics. *Com. Rev. 7*, 4 (Jul.–Aug. 1966), B-1 to B-31.
23. Slagle, J. R. A heuristic program that solves symbolic integration problems in freshman calculus. In *Computers and Thought*, pp. 191–203. CR-6236-3068.

Pattern recognition and picture processing [Item 5]

24. McCormick, B. H., Ray, S. R., Smith, K. C., and Yamada, S. ILLIAC III: A processor of visual information. Proc. IFIP Congress, New York, 1965, Vol. 2, Spartan Books, New York, pp. 359–361.
25. Tippett, J. T., Berkowitz, D. A., Clapp, L. C., Koester, C. J., and Vanderburgh, A., Jr. (Eds.) *Optical and Electro-Optical Information Processing*, Proc. Symp. Optical and Electro-Optical Inf. Proc. Tech., Boston, Nov. 1964. M.I.T. Press, Cambridge, Mass., 1965, 780 pp. CR-6673-9829.
26. Uhr, L. (Ed.) *Pattern Recognition.* Wiley, New York, 1966, 393 pp. CR-6674-10,028.

General problem solver and advice taker [Item 6]

27. McCarthy, J. Programs with common sense. In D. V. Blake and A. M. Uttley (Eds.), *Proc. Symp. on Mechanisation of Thought Processes*, Two volumes, National Physical Laboratory, Teddington, England. H.M. Stationery Office, London, 1959, pp. 75–84.
28. Newell, A., Shaw, J. C., and Simon, H. A. A variety of intelligent learning in a general problem solver. In M. Yovits and S. Cameron (Eds.), *Self-Organizing Systems*, Pergamon Press, New York, 1960, pp. 153–159. CR-6236-2908.
29. Newell, A., and Simon, H. A. Computer simulation of human thinking. *Science 134*, 3495 (22 Dec. 1960), 2011–2017. CR-6234-2062.

Question answering programs [Item 7]

30. BOBROW, D. G. A question answering system for high school algebra word problems. Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 591–614. CR-6562-7183.
31. GREEN, B. F., WOLF, A. K., CHOMSKY, C., AND LAUGHERY, K. Baseball: an automatic question answerer. In *Computers and Thought*, pp. 207–216. CR-6341-3417.
32. LINDSAY, R. K. Inferential memory as the basis of machines which understand natural language. In *Computers and Thought*, pp. 217–233.
33. RAPHAEL, B. A computer program which "understands." Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Spartan Books, New York, pp. 577–589. CR-6562-7207.
34. SIMMONS, R. F. Answering English questions by computer—a survey. *Comm. ACM 8*, 1 (Jan. 1965), 53–70. CR-6563-7643.

Verbal and concept learning [Item 8]

35. FEIGENBAUM, E. A. The simulation of verbal learning behavior. In *Computers and Thought*, pp. 297–309. CR-6234-2060.
36. FEIGENBAUM, E. A., AND SIMON, H. A. Forgetting in an associative memory. Preprints of papers presented at the 16th Nat. Meeting of the ACM, Los Angeles, Sept. 5–8, 1961, Association for Computing Machinery, New York. CR-6232-1667.
37. HUNT, E. B. *Concept Learning: An Information Processing Problem*. Wiley, New York, 1962, 286 pp. CR-6561-6872.
38. MILLER, G. A., GALANTER, E., AND PRIBRAM, K. *Plans and the Structure of Behavior*. Holt, Rinehart and Winston, New York, 1960.

Decision making programs [Item 9]

39. CLARKSON, G. P. E. A model of the trust investment process. In *Computers and Thought*, pp. 347–371. CR-6563-7473.
40. FELDMAN, J. Simulation of behavior in the binary choice experiment. In *Computers and Thought*, pp. 329–346. CR-6342-3760.
41. FINDLER, N. V. Human decision making under uncertainty and risk: computer-based experiments and a heuristic simulation program. Proc. AFIPS 1965 Fall Joint Comput. Conf., Pt. I. Spartan Books, New York, pp. 737–752. CR-6673-9594.

Music composition [Item 10]

42. Computers in Music. Session 7, Tues. Nov. 8, at the AFIPS 1966 Fall Joint Computer Conf., San Francisco. (The papers for this session were not published in the conference proceedings.)
43. GILL, S. A technique for the composition of music in a computer. *Comput. J. 6*, 2 (July 1963), 129–133. CR-6451-4983.
44. HILLER, L. A., JR., AND ISAACSON, L. M. *Experimental Music*. McGraw-Hill, New York, 1959, 197 pp. CR-6012-0047.
45. MATHEWS, M. V. The digital computer as a musical instrument. *Science 142*, 3592 (1 Nov. 1963), 553–557.
46. REITMAN, W. R. *Cognition and Thought: An Information Processing Approach*. (Chap. 6). Wiley, New York, 1965, 312 pp.
47. SEAY, A. The composer of music and the computer. *Comput. Autom. 13*, 8 (Aug. 1964), 16–18. CR-6563-7548.

Learning nets and neural networks [Item 11]

48. ARBIB, M. *Brains, Machines and Mathematics*. McGraw-Hill, New York, 1964, 163 pp. CR-6455-6254.
49. BLOCK, H. D. Adaptive neural networks as brain models. *Experimental Arithmetic, High Speed Computing and Mathematics*, Proc. of Symposia in Appl. Math. 15, American Mathematical Society, Providence, R. I., 1963, pp. 59–72. CR-6453-5608.
50. LETTVIN, J. Y., MATURANA, H., MCCULLOCH, W. S., AND PITTS, W. What the frog's eye tells the frog's brain. *Proc. IRE 47*, (1959), 1940–1951.
51. ROSENBLATT, F. *Principles of Neurodynamics*. Cornell Aeronaut. Lab. Rep. 1196-G-8, Spartan Books, New York, 1962.
52. YOUNG, J. Z. *A Model of the Brain*. Clarendon Press, Oxford, England, 1964, 384 pp.

Adaptive systems [Item 12]

53. FOGEL, L. J., OWENS, A. J., AND WALSH, M. J. *Artificial Intelligence Through Simulated Evolution*. Wiley, New York, 1966, 170 pp.
54. NILSSON, N. J. *Learning Machines*. McGraw-Hill, New York, 1965, 137 pp. CR-6565-8177.
55. TOU, J. T., AND WILCOX, R. H. (Eds.) *Computer and Information Sciences*. Proc. of Symposium at Northwestern University, 1963, Spartan Books, New York, 1964, 544 pp.
56. VON FOERSTER, H., AND ZOPF, G. W., JR., (Eds.) *Principles of Self-Organization*. Pergamon Press, New York, 1962.
57. YOVITS, M. C., JACOBI, G. T., AND GOLDSTEIN, G. D. (Eds.) *Self-Organizing Systems, 1962*. Spartan Books, New York, 1962, 563 pp. CR-6456-6603.

Natural language processing [Item 13]

58. BAR-HILLEL, Y. *Language and Information: Selected Essays on Their Theory and Application*. Addison-Wesley, Reading, Mass., 1964, 388 pp. CR-6562-7178.
59. BOBROW, D. G. Syntactic analysis of English by computer—a survey. Proc. AFIPS 1963 Fall Joint Comput. Conf., Vol. 24, Spartan Books, New York, pp. 365–387. CR-6671-8838.
60. CHOMSKY, N. *Aspects of the Theory of Syntax*. M.I.T. Press, Cambridge, Mass., 1965, 251 pp. CR-6676-10,735.
61. GARVIN, P. L. (Ed.) *Natural Language and the Computer*. McGraw-Hill, New York, 1963, 398 pp. CR-6456-6569.
62. HAYS, D. (Ed.) *Readings in Automatic Language Processing*. American Elsevier, New York, 1966, 202 pp.

Questions of philosophical import [Item 14]

63. MACKAY, D. M. Mind-like behavior in artifacts. *Brit. J. Phil. Sci. 2*, (1951), 105–121.
64. SAYRE, K. M., AND CROSSON, F. J. (Eds.) *The Modeling of Mind: Computers and Intelligence*. University of Notre Dame Press, Notre Dame, Ind., 1963, 275 pp. CR-6455-6205.
65. SIMON, H. A. The architecture of complexity. *Proc. Am. Phil. Soc. 106*, (1962), 467–482.
66. TURING, A. M. Computing machinery and intelligence. In *Computers and Thought*, pp. 11–35.