## Tutorial Week 80

**Final Program**

A solid week of tutorials
December 15–19, 1980

**HOTEL DEL CORONADO • SAN DIEGO, CALIFORNIA**

<table>
<thead>
<tr>
<th>Monday, 12/15</th>
<th>Tuesday, 12/16</th>
<th>Wednesday, 12/17</th>
<th>Thursday, 12/18</th>
<th>Friday, 12/19</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 VLSI from a User’s Perspective&lt;br&gt;Rex Rice</td>
<td>2 Design of Microprocessor Systems&lt;br&gt;John H. Carson</td>
<td>3 Microprocessor Interfacing&lt;br&gt;G. Jack Lipovski</td>
<td>4 Microcomputer Control System Programming (Repeated as Tutorial No. 17)&lt;br&gt;Lance A. Leventhal</td>
<td>5 Microcomputer System Software and Languages&lt;br&gt;Belton E. Allen</td>
</tr>
<tr>
<td>6 Distributed Processing Systems&lt;br&gt;Kenneth J. Thurber</td>
<td>7 Distributed Processor Communication Architecture&lt;br&gt;Kenneth J. Thurber</td>
<td>8 Local Computer Networks&lt;br&gt;Harvey A. Freeman</td>
<td>9 Distributed System Design&lt;br&gt;Michael P. Mariani and David F. Palmer</td>
<td>10 Distributed Data Base Systems&lt;br&gt;Carl Cagan</td>
</tr>
<tr>
<td>11 Software Management I&lt;br&gt;Donald J. Reifer</td>
<td>12 Software Management II&lt;br&gt;Donald J. Reifer</td>
<td>13 Requirements Analysis and Definition&lt;br&gt;Peter Freeman and Anthony I. Wasserman</td>
<td>14 Software Design Techniques&lt;br&gt;Peter Freeman and Anthony I. Wasserman</td>
<td>15 Systematic Software Construction&lt;br&gt;Peter Freeman and Anthony I. Wasserman</td>
</tr>
<tr>
<td>16 An Intelligent Spouse’s Guide to Computers&lt;br&gt;(3 special half-day sessions for novices)&lt;br&gt;Ware Myers&lt;br&gt;Part I&lt;br&gt;Part II&lt;br&gt;Part III</td>
<td>17 Microcomputer Control System Programming (Tutorial No. 4 Repeated)&lt;br&gt;Lance A. Leventhal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
I actively dautomation.

**Rex Rice** is an independent consulting engineer specializing in digital systems, and has been actively engaged in the computing field since 1949. He has published over 50 papers and holds more than 20 computer patents. While at IBM he managed the development of the first transistorized computer.

He is best known for his work in Fairchild, the invention of the Dual In-Line Package (DIP); managing the first large silicon memory program, the ILLIAC IV main memory; and for his early perception of work in bus-oriented, functionally split array logic systems. The Symbolic computer which he conceived and the program he managed (1966–70) to develop it, have provided the most sophisticated direct implementation of a high-level language in hardware presently in existence. Many of its features are just now appearing in new systems.

Rice, who received his BA in mechanical engineering at Stanford in 1940, has taught graduate computer courses at USC. He is an IEEE Fellow, a member of the Computer Society Governing Board, and founder of the COMPACOM.

Course Outline: Historical Review and Attempted Forecast: Steps from transistors to VLSI will be traced. Emphasis will be placed on economic motivations and changing job skills for workers in the field.

Current Examples: One university and two commercial examples of successful VLSI programs will be reviewed. Two interesting but less successful projects will be covered for contrast.

Description of VLSI Design, Fabrication and Testing Process: Each step in the process will be discussed covering critical parameters and required decisions: choice of circuit/silicon technology • system architecture/logic simulation • design automation • marketing, fabrication and assembly • testing

Hazards Frequently Encountered Economic Review and Wrap-Up

**Tutorial No. 3**

**Microprocessor Interfacing**

Instructor: G. Jack Lipovski

**Audience:** Microcomputer system designers, engineers who interconnect microprocessors with external equipment, programmers who write software for interface hardware. Some experience in logic design and in assembler language on some machine is required.

**Course Description:** Emphasizes the principles of interface design and shows applications of interface design using the Intel 8048 and hex, and some compatible Motorola 6800 inputs–output chips. Topics covered include: (1) good programming practice for interface programs, subroutines, data structures and arithmetic, and the design of buses and address decoders; (2) the hardware design and software control of simple parallel I/O devices and interrupts; (3) a survey of conventional analog circuits used in microcomputers and discussion of frequency modulation analog circuitry; and (4) communications circuitry and programming.

G. Jack Lipovski is an associate professor of electrical engineering and computer science at the University of Texas at Austin, where he teaches microcomputer interfacing and computer architecture. While at the University of Texas, he designed the Texas Reconfigurable Array Computer. Earlier, at the University of Florida, he designed the CASSM data base computer. Dr. Lipovski is a past chairman of the Technical Committee on Computer Architecture of the Computer Society of the IEEE.

He is a member of the Computer Society Governing Board and a Director of Euromicro.

**Course Outline:**

Introduction: the M6809 instruction set • data structures • subroutines • computer arithmetic • bus timing • address decoding

Parallel I/O: simple I/O • a traffic light controller • M6821 (FIA) • I/O • an I/O tester • real time and gaddly synchronization interrupts

**Analog Devices:** transducers • OP amps • hardware analog/digital and software tradeoffs • D-to-A converters • A-to-D converters • counters • M6840 • period measurement

**Communications Devices:** levels of protocol • RS232 standard • modems • the UART • the M6850 (ACIA) • 5680LC and IEEE-488 protocols
Microcomputer Control System Programming

Instructor: Lance A. Leventhal

Audience: Intended for design engineers, programmers, and engineering managers; also useful for programmers with mini or large computer experience.

Course Description: Reviews the basic problems in microcomputer-based design and the basic tasks involved in such designs. Introduces the Intel 8085 microprocessor and the Intel SDK-85 microcomputer. Students will then undertake a series of hands-on experiments in microcomputer operation and programming. The experiments will cover examining and changing memory, executing programs, writing and running simple programs, handling switch inputs, examining communications lines, handling LED outputs, writing delay routines, and establishing output duty cycles. Experiments and problems are drawn from the design of controllers for instruments, test equipment, communications equipment, peripherals, business equipment, factories, chemical processes, navigation and guidance systems, and consumer electronics. The laboratory manual describes reference materials, defines terms and instructions, and provides complete examples that are ready to run.

Lance A. Leventhal is an independent consultant specializing in hardware and microprogramming. He also serves as technical editor for the Society for Computer Simulation and as contributing editor for Digital Design magazine. He is the author of 10 books and over 60 articles on microprocessors, and in is a regular contributor to such publications as Simulation, Digital Design, and Microcomputing. He received his PhD in applied physics and information science from the University of California at San Diego.

Course Outline:

Current Situation in Microprocessors: requirements, costs, key problems, useful approaches

Intel 8085 Microprocessors: architecture, registers, arithmetic facilities, instruction set

Intel SDK-85 Microcomputer: addresses, memory examining and change, program execution, register examination and change, single-step mode

Introduction to the SDK-85 Microcomputer: overview, resetting the computer, examining memory, changing memory, executing a program

Writing and Running Simple Programs: one complement program, using registers H and L, examining registers, comparing programs, adding two numbers

Simple Input: I/O instructions, SDK-85 input, flags and conditional jumps, waiting for a switch closure, special bit positions, examining flags, waiting for two closures, search for a starting character

Simple Output: 8355 I/O ports, turning on an LED, providing a delay, controlling individual bits, establishing a cycle

Future Trends: applications, hardware software, overall significance

Microcomputer System Software and Languages

Instructor: Belton E. Allen

Audience: Design engineers, programmers, system programmers, and technical managers involved in specifying, designing, or implementing development and support systems for microprocessor-based products. Prospective attendees are assumed to have a basic knowledge of microcomputer software development.

Course Description: Examines the software development process for microprocessor-based products, with emphasis on understanding the various development tools and their application. Presents tradeoffs for resident versus cross development and for high-level versus assembly language. A system designed to simultaneously support software development for microprocessors from various manufacturers will be discussed in detail. Practical examples will be drawn from real-life microprocessor-based products.

Belton E. Allen is the software director of Alcyon Corporation, a manufacturer of microcomputer systems. His work involves the design and implementation of system software and support tools. He was previously assistant professor of computer science at the Naval Postgraduate School in Monterey, California, where he led a research project on software development support. He holds a BS in mathematics from Rice University and an MS in computer science from the University of Houston.

Course Outline:

Overview of Microcomputer Software Development Process

Microcomputer Software Development Tools: text editor, filing system, assembler, compiler, linking loader, debugging tool, in-circuit emulator

Review of Commercially Available Development Systems: resident vs. cross development

Programming Language Choices: assembly vs. high-level, compiler vs. interpreter, which high-level language?

A Case Study of a Minicomputer-Based Development System for Support of Multiple Microprocessors

Impact of the 16-Bit Microprocessors on the Software Development Process

Distributed Processing Systems

Instructor: Kenneth J. Thurber

Audience: Managers, engineers, programmers, and systems analysts.

Course Description: Provides an overview of current concepts and problems associated with implementing a distributed processing system. Defines the elements of a distributed system and cites contemporary models of such systems. It describes three case studies: ETHERNET, SNA, and HYPERChannel, and concludes with a survey of design issues associated with system software, interconnection systems, topology, and hardware concepts.

Kenneth J. Thurber is a senior staff scientist with Sperry Unisys. He is the author of several textbooks and over 50 technical papers. His interests include local networks, communication architectures, distributed processing systems, computer architecture, and techniques for requirement analysis. He is currently chairman of the Computer Society's Technical Committee on Computer Communications, and is a Distinguished Visitor of the Computer Society. He was previously employed by Honeywell, and is currently a part-time faculty member of the Computer Science Department of the University of Minnesota. Thurber received a PhD in electrical engineering from Montana State University in 1969. He is the general editor of the Computer Science Series published by the Lexington Books Division of D. C. Heath.

Course Outline:

Introduction: overview, nomenclature, reasons for distributed processing, networks, applications

Models: virtual circuits, datagrams, standards, hierarchical systems, common issues, system approaches, topology, multiplexing

Design Issues: operating systems, protocol hierarchies, gateways, communication subsystems, contention channels, process partitioning

Case Studies: ETHERNET, SNA, HYPERChannel, virtual machine networks, future systems, concluding remarks

All tutorials start at 9 a.m.
TUTORIAL NO. 7

Distributed Processor Communication Architecture

Instructor: Kenneth J. Thurber

Audience: Managers, engineers, programmers, and systems analysts.

Course Description: This introductory tutorial introduces data transfer mechanisms used in distributed processing and networks. Areas of discussion include functional topology issues of distributed and network systems; implementation of data transfer and route, addressing techniques, and functional aspects of network systems.

Kenneth J. Thurber is a senior staff scientist with Sperry Univac. He is the author of several textbooks and over 50 technical papers. His interests include local networks, communication architectures, distributed processing systems, computer architecture, and techniques for requirement analysis. He is currently a member of the Computer Society's Technical Committee on Computer Communications, and is a Distinguished Visitor of the Computer Society. He was previously employed by Honeywell and is currently a part-time faculty member of the computer science department at the University of Illinois. He is the general editor of the Computer Science Series published by the Lexington Books Division of D. C. Heath.

Course Outline:

Introduction and Systems: overview definitions models networks Anderson/Jensen taxonomy applications

Packet Switched Networks: definitions traffic types hierarchy concepts design issues protocols layered structure distributed routing techniques centralized routing techniques flow control segmentation and addressing error control name management

Circuit Switching Systems: connection networks space division multiplexers time division multiplexers construction techniques concentrators expanders superconcentrators Banyan structures Omega structures

Bus Structures and Conclusions: bus models bus taxonomy bus arbitration synchronous communication synchronous communication information transfer bus width transmission issues

TUTORIAL NO. 8

Local Computer Networks

Instructor: Harvey A. Freeman

Audience: Intended for managers, engineers, programmers, system analysts, designers, and students. Some experience in computing and some exposure to data communications would be helpful.

Course Description: This introductory level tutorial addresses local computer networks, a class of systems which are generally owned by a single organization, cover distances on the order of a few miles, and use communication sub-network technology such as packet switching, circuit switching, or busing. The overall design and content of LCNs are described. Then important design issues and potential solutions currently identified for LCNs are discussed. This is followed by a number of case studies of important local computer networks. LCNs which evolved from distributed processing contexts and those which evolved from attempts to upgrade and enhance current large-scale systems are covered. Concludes with a view of the future directions of LCNs.

Harvey A. Freeman is manager of the Data Systems Design Department at Sperry Univac. He currently manages an advanced development group investigating and developing concepts, techniques, and specifications for large-scale Sperry Univac computer systems in distributed processing environments, including local, back-end, and non-homogeneous networks. He is the author of numerous technical papers and co-author of the book, Data Base Computers. Freeman is also an adjunct professor in the computer science department at the University of Minnesota. He holds a PhD in electrical engineering from the University of Illinois.

Course Outline:

Background: definition evolution examples

Design Issues: network entities network architectures network operations network performance user considerations

Case Studies: HYPERchannel ETHERNET Cambridge Ring selected others

Future Directions: gateways commercial availability trends

Conclusions

TUTORIAL NO. 9

Distributed System Design

Instructors: Michael P. Mariani and David F. Palmer

Audience: Intended for data processing systems engineers and designers. Also of interest to first and second level technical managers. Not intended for detailed hardware designers.

Course Description: We may approach the design of a distributed computing system with enthusiasm. The opportunity of molding custom architectures of hardware and software components to attain enhanced throughputs, reliability, "changeability," etc., inspires our artful selves. The many options for exploiting parallelism, concurrency, redundancy, modularity, and design management can quickly become impossible as the project size increases.

This tutorial provides a procedure for treating design decisions in an orderly, visible manner. It describes techniques and computer aids applicable to specific steps of the procedure. Geographical, local systems, and elemental component distributions are treated. Examples are drawn from stressing, real-time systems, and multicomp computer tests simulations.

Michael P. Mariani is program manager for TRW's Distributed Processing Architecture Design Program, which is directed at establishing the architecture technology necessary for low-risk design and development of stringent real-time data-driven systems. Mariani is an IEEE Distributed Systems Design Lecturer for 1979-1980 and a member of IFIP TC 10.3 Working Group on Digital Systems Design. He is a graduate of the University of Illinois.

David F. Palmer is manager of the Distributed Processing Requirements Engineering Program, General Research Corporation, where he has been developing requirements for software hardware tradeoff definitions leading to customized distributed architectures. He has taught courses in computer engineering and computer science at the University of California, and is an IEEE Computer Society Distinguished Lecturer for 1979-80. Palmer holds a PhD in electrical engineering from Duke University.

Course Outline:

Distributed System Design Rationale

Analysis Steps: identification elaboration operational requirements accommodation verification

Partitioning Steps: criteria selection relationship evaluation thresholding

Allocation Steps: architecture dimensionality determination module mapping resource recommendations performance-cost-reliability analysis

Synthesis Steps: interconnect architecture design high fidelity performance analysis pay-off models analytic contention models discrete event simulations

Summary and Conclusions

All tutorials start at 9 a.m.
TUTORIAL NO. 10

Distributed Data Base Systems
Instructor: Carl Cagan

Audience: Programmers, analysts, system designers, and data base administrators who have several years of experience in computing. Assumes exposure to data base applications and data base management systems.

Course Description: Describes distributed data base systems in operation and in development. Provides coverage of current directions for solutions to problems engendered by distributive qualities (e.g., query processing, redundant update, heterogeneity).

Carl Cagan is a senior computer scientist at Computer Sciences Corporation. He has 25 years of experience in the computer field including positions as associate professor and director of the Computer Center at California State College, Dominguez Hills, as an independent consultant, and as a staff member at Hughes Aircraft Co., SDC, Informatics, and Ramo-Wooldridge. His interests include data base software, data base machine technology, and information retrieval systems.

Cagan is the author of Data Management Systems and of several articles on information retrieval. He received the MS and PhD in computer science from Washington State University.

Course Outline:
- Data Models and Structures
- Data Base Machine Overview
- Directory Storage and Processing
- Query Processing
- Concurrency Control: synchronization * deadlock
- Data Protection and Recovery
- Distributed Data Base Systems: in development * in production

TUTORIALS NO. 11 & 12

Software Management I & II
Instructor: Donald J. Reifer

Audience: Intended for project managers, group supervisors, line managers, and senior staff interested in learning how to manage software projects and personnel. Assumes no extensive training or experience in software.

Course Description: Provides a basic understanding of the skills of software management. Focuses on the basic functions involved in managing a medium- to large-scale programming project in data processing operations: planning, organizing, staffing, directing, and controlling. Introduces each of the functions and provides insight via the case method into the practical use of the key theories, methods, tools, and techniques available. Both commercial and military applications are discussed, as are the tradeoffs between alternative approaches. Examples illustrating approaches for both large and small systems are presented.

Donald J. Reifer is president of Software Management Consultants, a consulting firm specializing in software management-related services. He works with clients to identify technical and programmatic risks and recommends action plans to reduce their consequences. While affiliated with the Aerospace Corporation, he managed all the software activities conducted in support of the Space Shuttle. These involved managing multiple projects producing over $100 million in software deliverables. Earlier, at Hughes Aircraft, he was responsible for factory automation, simulation, and software research projects. He has been an elected member of the General Accounting Office and the National Science Foundation, and has been asked to testify before Congress.

Reifer received a BSEE from Newark College of Engineering, an MS in operations research from USC, and the Certificate in Business Management for Technical Personnel from UCLA.

Course Outline
- Introduction: setting the stage - challenges and issues * software management and the system life cycle * the manager's job * do's and don'ts for project managers * getting the customer involved early * requirements definition summary * evaluation of automated requirements systems
- Planning: the need for policy * program definition and the project plan * selecting the software engineering approach * planning tools and techniques * milestone definition and scheduling * cost estimation guidelines * evaluation of software cost models * risk reduction techniques
- Organizing: organizational options * effective use of matrix management * conflict resolution * team approaches and building
- Staffing: organizing and recruiting the project plan * management by objectives * staffing the project plan * the computer science engineering dilemma
- Direction: delegating and passing the buck * motivating software professionals * thoughts on leadership
- Controlling: measuring project performance * overcoming the 90% complete syndrome * configuration management * quality assurance * effective audit and review techniques

Summary
- Question and Answer Session

TUTORIAL NO. 13

Requirements Analysis and Definition
Instructor: Peter Freeman and Anthony I. Wasserman

Audience: Programmers, analysts, designers, and managers of software.

Course Description: Introduces the basic concepts underlying any systematic definition of software system requirements, and then surveys several specific techniques that employ these concepts, including SADT, structured systems analysis, RSL, and various specification techniques. Also treated is the relationship between requirements and specifications, some of the human factors involved in requirements definition, and management considerations.

(To be taken in conjunction with Tutorials No. 14 and 15)

Peter Freeman is an associate professor of information and computer science at the University of California, Irvine. He has been involved in the analysis, design, and construction of advanced computer systems and the training of software engineers since 1961.

Freeman's research activities have been concentrated in computer assisted software engineering and their application to the software engineering process. He is active in professional organizations and is well known for his seminars on software design and software engineering. He has served as a consultant to the United Nations, the U.S. government, and numerous industrial organizations.

Freeman has published numerous technical papers and is the author of Software Systems Principles (SRA, 1975). In addition, he has jointly edited (with Prof. Wasserman) two books: Software Engineering Education (Springer-Verlag, 1976) and Tutorial on Software Design Techniques (IEEE Computer Society, 1980). He received his PhD in computer science from Carnegie-Mellon University in 1970.

Anthony I. Wasserman (See Tutorial No. 14.)

Course Outline:
- Basic Concepts of Requirements Analysis and Definition (RA&D): historical background and goals * role in the life cycle * work products of RA&D * basic processes
- Techniques Used for RA&D: SADT, structured systems analysis, RSL, and other techniques will be surveyed, presenting basic concepts, examples, and current usage
- Relationship Between Requirements and Specifications: requirements relate to needs * specifications describe something to be built * requirements analysis results in a specification
- Human Factors in RA&D: need for communication * communication paths * deterrents to communication
- Management of RA&D: planning the analysis activity * keeping track of progress * teamwork * assuring quality
- Developing a Methodology for RA&D: typical problems in changing methods * a strategy for overcoming them
- Overview of Current Research in RA&D

All tutorials start at 9 a.m.
**Tutorial NO. 14**

**Software Design Techniques**

**Instructors:** Peter Freeman and Anthony I. Wasserman

**Audience:** Programmers, analysts, designers and managers of software.

**Course Description:** Provides a balanced introduction to most modern software design techniques. Treats concepts, explicit methods, and current practice in the areas of design strategy, architectural and detailed design, design quality, and management of the design activity. Provides a unique approach to understanding the continually expanding repertoire of design methods by stressing the intellectual basis of the design activity. 

*(To be taken in conjunction with Tutorials No. 13 and 15)*

Peter Freeman (See Tutorial No. 13.)

Anthony I. Wasserman is an associate professor of medical information science at the University of California, San Francisco, and a lecturer in the Computer Sciences Division at the University of California, Berkeley. His research interests include programming language design, programming methodology, database management systems, computer science education, and applications to health care.

Wasserman has been active in professional organizations, presently serving as chairman of ACM SIGSOFT, the Special Interest Committee on Software Engineering. He has served as an ACM National Lecturer and was the U.S. conference chairman for the 4th International Conference on Very Large Data Bases. He is the author of over 20 technical papers, co-editor of Software Engineering Education - Needs and Objectives, and is revising Concepts of Programming Languages written by the late Mark Elson.

Wasserman received his PhD in computer science from the University of Wisconsin, Madison, and his AB in mathematics and physics from the University of California, Berkeley.

**Course Outline:**

**Introduction:** software engineering and the role of design • the interplay of system and data design • payoffs from improved design

**Design Theory:** fundamental operations of design • the role of representation, experience, and knowledge • desired characteristics of the design product and methods

**Specifications:** goals for specifications • formal specification techniques

**Architectural Design:** importance of structure • abstraction • modularity and structural models

**Survey of Structuring Methods:** SADT, Parnas, Petri nets, FSM, structured design

**Detailed Design Program Design**

**Languages Survey:** LCP • step-wise refinement • Jackson

**Design Quality:** establishing quality expectations • testing design • complexity measures and their use

**Design Management:** allocating and controlling design resources • design organizations and reviews

---

**Tutorial NO. 15**

**Systematic Software Construction**

**Instructors:** Peter Freeman and Anthony I. Wasserman

**Audience:** Programmers, analysts, designers, and managers of software.

**Course Description:** Surveys current concepts and techniques for systematic software construction. Begins with a review of detailed design and the development life cycle. Illustrates and explains objectives and features of structured coding, modern programming languages, and programming style. Presents the concepts and the current technology of programming environments and tools, as well as development strategy and programming management techniques.

*(To be taken in conjunction with Tutorials No. 13 and 14)*

Peter Freeman (See Tutorial No. 13.)

Anthony I. Wasserman (See Tutorial No. 14.)

**Course Outline:**

**Implementation and the Software Life Cycle:** the role of implementation • preceeding phases • design vs. programming • the programming environment

**Review of Detailed Design:** program design languages • flowcharts • structured and unstructured • stepwise refinement

**Structured Programming:** historical background • the GOTO controversy • the role of testing and verification • implications for languages and style

**Modern Programming Languages:** Pascal and its descendants • control structures • data types and data abstractions • modularization

**Programming Style:** good programs in "bad" languages and vice versa • readability • avoidance of programming tricks • restricted use of local data • information hiding • control flow restrictions • static vs. dynamic data structures

**The Programming Environment:** tools • interactive development • coherent methodologies and their tools

**Development Strategies:** top down vs. bottom up • coding and unit testing • correctness and performance

**Management and System Construction:** organization of programming projects • configuration management • program production libraries • structured walkthrough and code reviews

---

**Tutorial NO. 16**

**An Intelligent Spouse's Guide to Computers**

**Instructor:** Ward Myers

**Audience:** Planned for persons with no education or experience in the computer field. There are no prerequisites except an interest in finding out enough about computers to appreciate their role in today's society. Attendees may be the spouses of computer professionals who seek more understanding of their partner's work.

**Course Description:** A computer is fundamentally simple: it takes an instruction from its memory and performs the arithmetic or logical operation the instruction specifies. Creating a string of these instructions is called programming. Computer programming is based on only a few simple steps, and these simple steps are carried out millions of times. Our purpose is to grasp the underlying simplicities and on this foundation come to appreciate what the computer can do and what effect it is having on us.

We will be assisted in this purpose by two or three 15-minute video tapes each day. Prepared by Evolution I, a division of Electronic Data Systems Corp., these programs help us visualize and understand the world of computers.

**Ward Myers** is an engineer, a teacher, and a writer. He is a contributing editor for *Computer* magazine and has published widely in trade and professional publications. Although he has explored many facets of computers, he is not so much of a specialist that he can't be understood by the layman.

**Course Outline:**

**The Computer:** how a computer functions • comparison with the brain • patterns of problem solving • how the computer can and cannot be used only because electrical signals to the languages programmers use

**The Hardware:** how the central processor reads an instruction from its memory, performs it, and advances to the next instruction • how it stores data in magnetic tape recorders and disks

**Communications:** how computers communicate with each other • keyboards, printers, video screens, etc. • how they transmit information over communication channels • how they can be used to control a process

**Applications:** how the computer reproduces speech, recognizes speech, synthesizes music, and creates graphics on a TV screen

**Social Implications:** what effect computers will have on life in the next decade

---

**Tutorial Week**

Chairman: David Stornberg, CSC

Arrangements: Robert A. Kerr, CSC

Chairman, Tutorial Week West Standing Committee: Joseph P. Fernandez, IBM Corp.

Chairman, IEEE-CS Professional Development Committee: Stanley Winkler, IBM Corp.

All tutorials start at 9 a.m.
Tutorial Week '80
Advance Registration

Attendance limited to 100 per tutorial except for Tutorials No. 4 & 17, which are limited to 30 each. There will be no registration at the door. Advance registration only; deadline — November 24, 1980. Select up to one tutorial per individual per day. Requests for refunds must be received in writing by December 1, 1980.

FEES: (for all but Tutorial No. 16)*

<table>
<thead>
<tr>
<th></th>
<th>Per Tutorial</th>
<th>All Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Members</td>
<td>$85</td>
<td>$400</td>
</tr>
<tr>
<td>Non Members</td>
<td>$100</td>
<td>$475</td>
</tr>
</tbody>
</table>

Includes bound tutorial and luncheon. There will be a no-host reception Monday night.

MONDAY, 12/15

Tutorial No. 1
VLSI from a User’s Perspective

Tutorial No. 6
Distributed Processing Systems

Tutorial No. 11
Software Management I

Tutorial No. 16
An Intelligent Spouse’s Guide to Computers
(3 special half-day sessions for novices)

TUESDAY, 12/16

Tutorial No. 2
Design of Microprocessor Systems

Tutorial No. 7
Distributed Processor Communication Architecture

Tutorial No. 12
Software Management II

WEDNESDAY, 12/17

Tutorial No. 3
Microprocessor Interfacing

Tutorial No. 8
Local Computer Networks

Tutorial No. 13
Requirements Analysis and Definition

FRIDAY, 12/19

Tutorial No. 4
Microcomputer Control System Programming Limit — 30 People

Tutorial No. 5
Microcomputer System Software and Languages

Tutorial No. 10
Distributed Data Base Systems

Tutorial No. 15
Systematic Software Construction

Tutorial No. 17
Microcomputer Control System Programming Limit — 30 People

(Tutorial No. 4 Repeated)

* Special rate for Tutorial No. 16
(3 half-day sessions): $75 (includes luncheons)

I wish to register for the following tutorials (Please circle numbers):

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

☐ Check enclosed  ☐ Visa/BankAmericard  ☐ Master Charge

Card No._________________________Expiration Date__________Signature_______________________________

Name___________________________IEEE or IEEE-CS Membership No._____________________

Street Address___________________Phone No._____________________

City/State/Zip_____________________

MAIL TO:
Tutorial Week ’80
P.O. Box 639
Silver Spring, MD 20901

ROOM RESERVATION REQUEST
IEEE COMPUTER SOCIETY
December 15–19, 1980

Hotel del Coronado

Please check room preference

☐ Single or Double $36.75
☐ Lanai 63.75
☐ Parlor or Suite 101.25

Plus prevailing city occupancy tax per room per day.

I will arrive on (Date)______________Time__________

I will depart on (Date)______________

Please include ONE night’s advance room DEPOSIT to confirm the room reservation.

No reservations by telephone, please.

Please mail this form to: Reservations Manager
Hotel Del Coronado
1500 Orange Avenue
Coronado, CA 92118

Reservations must be received by NOVEMBER 14, 1980, in order to fulfill your room reservation request.
If you cannot attend Tutorial Week ’80, use this order form to purchase any of the tutorial texts listed below.

<table>
<thead>
<tr>
<th>ORDER NO.*</th>
<th>TUTORIAL TITLES</th>
<th>QTY</th>
<th>MEMBER PRICE**</th>
<th>NON-MEMBER PRICE</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>258</td>
<td>Distributed Processor Communication Architecture</td>
<td></td>
<td>$15.00</td>
<td>$20.00</td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>Design of Microprocessor Systems</td>
<td></td>
<td>$12.00</td>
<td>$16.00</td>
<td></td>
</tr>
<tr>
<td>267</td>
<td>Distributed System Design</td>
<td></td>
<td>$15.00</td>
<td>$20.00</td>
<td></td>
</tr>
<tr>
<td>261</td>
<td>Distributed Database Systems</td>
<td></td>
<td>$15.00</td>
<td>$20.00</td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>VLSI from a User’s Perspective</td>
<td></td>
<td>$16.50</td>
<td>$22.00</td>
<td></td>
</tr>
<tr>
<td>299</td>
<td>A Pragmatic View of Distributed Processing Systems</td>
<td></td>
<td>$16.50</td>
<td>$22.00</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>Software Requirements Analysis and Specification</td>
<td></td>
<td>$15.00</td>
<td>$20.00</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>Programming Language Design</td>
<td></td>
<td>$15.00</td>
<td>$20.00</td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>Local Computer Networks</td>
<td></td>
<td>$12.00</td>
<td>$16.00</td>
<td></td>
</tr>
<tr>
<td>219</td>
<td>Software Management</td>
<td></td>
<td>$12.00</td>
<td>$16.00</td>
<td></td>
</tr>
</tbody>
</table>

Charge Card No. Exp. Date Signature
Name (please print) Member No.
Address
City/State/Zip Country

IMPORTANT INFORMATION:
* To assure prompt processing of your order, be sure to enter the 3-digit number that appears ahead of each publication title.
** Remember, member rates apply on the first copy for personal use only. No returns or refunds after 60 days from date of shipment (90 days overseas).
*** All unit prices include postage for 4th class book rate (U.S., Canada and Mexico). For priority shipping, add $2.00 per book. Overseas mail shipped sea mail (7-8 weeks delivery). For airmail (1 week delivery), please add $10.00 per book.

SEND TO: IEEE Computer Society 5855 Naples Plaza, Suite 301 Long Beach, CA 90803 (213) 438-4958
PAYMENT (check one):
☑ Check Enclosed ☐ Bill Visa/BankAmericard
☐ Bill Master Charge

Foreign checks must be in US dollars drawn on a US bank.