

SENIOR YEAR		FALL	SPRING
EECE 295	Project Management for EECE	3	–
EECE 296	EECE Design	–	3
EECE 297	Senior Engineering Design Seminar	1	–
	Cmp E Program Electives †	3	3
	Liberal Arts Core	3	3
	Technical Electives	6	3
	Open Electives	–	3
		16	15

† Computer engineering majors are encouraged to take EECE 116 in the spring of their freshman year in lieu of MSE 150. MSE 150 may be taken in the sophomore year.

‡ As described in "Computer Engineering Degree Requirements" subsection 6. At least one design domain expertise (DE) course required prior to EECE 296.

CmpE 203–204. Independent Study. Readings or projects on basic topics in computer engineering or related fields under the supervision of staff. Consent of instructor required. No more than 6 hours may be applied towards graduation. [Variable credit: 1–3 each semester]

CmpE 291–292. Special Topics. [Variable credit: 1–3 each semester]

Computer Science

E

DEPARTMENT CHAIR Daniel M. Fleetwood

ASSOCIATE CHAIR Douglas C. Schmidt

DIRECTOR OF UNDERGRADUATE STUDIES Robert E. Bodenheimer, Jr.

DIRECTOR OF GRADUATE STUDIES Jeremy P. Spinrad

PROFESSORS EMERITI Patrick C. Fischer, William H. Rowan Jr.

PROFESSOR EMERITA Charlotte F. Fischer

PROFESSORS Gautam Biswas, Benoit M. Dawant, Lawrence W. Dowdy, J. Michael Fitzpatrick, Gábor Karsai, Douglas C. Schmidt, Janos Sztipanovits


ASSOCIATE PROFESSORS Robert E. Bodenheimer, Jr., Douglas H. Fisher, Stephen R. Schach, Jeremy P. Spinrad

ASSOCIATE PROFESSOR OF THE PRACTICE Gerald H. Roth

ASSISTANT PROFESSORS Julie A. Adams, Yi Cui, Aniruddha S. Gokhale, Xenofon D. Koutsoukos, Yuan Xue

RESEARCH ASSISTANT PROFESSOR Bradley A. Malin

LECTURER Julie L. Johnson

 THE program in Computer Science blends scientific and engineering principles, theoretical analysis, and actual computing experience to provide undergraduate students with a solid foundation in the discipline. Emphasis is on computing activities of both practical and intellectual interest, and on theoretical studies of efficient algorithms and the limits of computation. Computer facilities are available for class assignments, team projects, and individual studies. Students are challenged to seek original insights throughout their study. Working in teams, participating in summer internships, supporting student professional organizations, and developing interdisciplinary projects are strongly encouraged.

The computer science major provides an excellent background for medical studies, and the flexibility provided by its many open electives and broadening electives allows students to prepare for medical school while earning a degree in computer science with a normal load in four years. Interested students should discuss their plans with their computer science adviser in the fall of their first year.

In addition to Bachelor of Science, degrees of Master of Science, Master of Engineering, and Doctor of Philosophy are also awarded in Computer Science. Many students choose to double major in mathematics.

Undergraduate Honors Program. Students interested in the Honors Program should apply to the department chair. See the *Special Programs* chapter for general requirements of the professional Honors Program in computer science.

Curriculum Requirements

The B.S. degree in computer science requires a minimum of 122 hours, with distribution as follows:

1. Mathematics (16–22 hours). Required components:

(a) A Calculus sequence (7–12 hours).

Selected from the following:

–150a, 150b, 170, 175

–155a, 155b, 175

–205a, 205b

(b) Linear algebra (3–4 hours): 194, 196, 204, or 205a.

(c) Statistics/Probability (3 hours): 216, 218, or 247.

Elective course (3 hours):

Selected from: 198, 200, 208, 215, 219, 221, 223, 226,
247, 250, 253, 274, 275, 288.

2. Science (12 hours).

Selected from the following list. Each is a four credit hour lab course.

Students are required to take at least one two-course sequence.

–Biological Sciences (110a–110b and 111a–111b)

–Biological Sciences (100, 218, 219)

–Chemistry (102a and 104a, 102b and 104b)

–Earth and Environmental Sciences (101 and 111)

–Materials Science and Engineering 150

–Physics (116a-b and 118a-b)

Recommended: Chemistry 102a and 104a, Physics 116a-b.

3. Introduction to Engineering (3 hours). ES 140.

4. Writing Component (3 hours). ES 210W or one designated “W” course.

5. Liberal Arts Core (18 hours). To be selected from the approved lists (see Distribution Requirements, p. 504). Three hours may be in a technology and society elective.

6. Computer Science Core (29 hours).

–Software/Problem Solving: CS 101, CS 201, CS 251, and CS 270.

–Hardware/Systems: EECE 116, CS 231, and CS 281.

–Foundations: CS 212 and CS 250.

7. Computer Science Depth (18 hours). To be selected from Computer Science courses numbered CS 240 or higher, or from the following EECE courses: 253, 254, 271, 272, 273, or 276. At least one course (i.e., 3 hours) must be a designated project course selected from CS 258, 265, 269, 274, 276, 279, 282, 283, 284, or 285.

8. Broadening Electives (6 hours). Adviser approval must be obtained for broadening electives. In particular, broadening electives are to be used for further study in areas that enhance majors as computer scientists. Computer science and computer engineering courses may not be used as broadening electives. Additional electives from the liberal arts core, international studies, business related courses, enhanced technical electives, or courses that lead to a double major are especially encouraged.

9. Open Electives (11–17 hours).

(Note: In the event that a given course could be used to satisfy, or partially satisfy, requirements in more than one of the above categories, the student and adviser must choose the category to which the course will apply. That is, no course may be "double-counted.")

Pass-Fail Courses. The only courses that computer science students may choose to take pass-fail are those in items 5, 8, and 9 above.

Specimen Curriculum for Computer Science

		Semester hours	
		FALL	SPRING
FRESHMAN YEAR			
Chem 102a	General Chemistry	3	–
Chem 104a	General Chemistry Laboratory	1	–
Physics 116a	General Physics	–	3
Physics 118a	General Physics Laboratory	–	1
Math 155a	Accelerated Single-Variable Calculus I	4	–
Math 155b	Accelerated Single-Variable Calculus II	–	4
ES 140	Introduction to Engineering	3	–
CS 101	Programming and Problem Solving	–	3
	Open Electives	3	–
	Liberal Arts Core	–	3
		14	14
SOPHOMORE YEAR			
Physics 116b	General Physics	3	–
Physics 118b	General Physics Laboratory	1	–
Math 175	Multivariable Calculus	–	3
EECE 116/116L	Digital Logic	4	–
CS 201	Program Design and Data Structures	3	–
CS 212	Discrete Structures	–	3
CS 231	Computer Organization	–	3
CS 251	Intermediate Software Design	–	3
	Liberal Arts Core	–	3
	Open Elective	3	–
		14	15

JUNIOR YEAR		FALL	SPRING
Math 194	Methods of Linear Algebra	–	3
Math 218	Introduction to Math Statistics	3	–
ES 210W	Technical Communications	3	–
CS 250	Algorithms	3	–
CS 270	Programming Languages	–	4
CS 281	Operating Systems Principles	3	–
	Computer Science Project	–	3
	Liberal Arts Core	3	3
	Computer Science Depth	3	3
		18	16
SENIOR YEAR			
	Math Elective (e.g., Math 250)	3	–
	Computer Science Depth	3	6
	Liberal Arts Core	3	3
	Broadening Electives	3	3
	Open Electives	4	3
		16	15

Second Major in Computer Science for Non-Engineering Students

The second major in computer science for students enrolled outside the School of Engineering requires 47 hours distributed according to items 6 and 7 of the *Curriculum Requirements* listed above.

Courses taken toward the second major may not be taken pass/fail.

Computer Science Minor

The minor in computer science requires 22 hours of computer science courses as follows:

- | | |
|--|----------|
| 1. Programming: CS 101 | 3 |
| 2. Discrete Structures: CS 212 | 3 |
| 3. Digital Logic Fundamentals: EECE 116/116L | 4 |
| 4. Intermediate Computer Concepts: CS 201 and 231 | 6 |
| 5. Two additional CS courses numbered 250 or above | <u>6</u> |

Total Hours: 22

Courses taken toward the minor may not be taken pass/fail.

CS 101. Programming and Problem Solving. An intensive introduction to algorithm development and problem solving on the computer. Intended for engineering majors and others who already have some familiarity with computer programming. Structured problem definition, top down and modular algorithm design. Running, debugging, and testing programs. Program documentation. FALL, SPRING. [3]

CS 103. Introductory Programming for Engineers and Scientists. An introduction to problem solving on the computer. Intended for students other than computer science and computer engineering majors. Methods for designing programs to solve engineering and science problems. Generic programming concepts. SPRING. [3]

CS 151. Computers and Ethics. Analysis and discussion of problems created for society by computers, and how these problems pose ethical dilemmas to both computer professionals

and computer users. Topics include: computer crime, viruses, software theft, ethical implications of life-critical systems. Technology-society elective. FALL, SPRING. [3]

CS 201. Program Design and Data Structures. Continuation of CS 101. The study of elementary data structures, their associated algorithms and their application in problems; rigorous development of programming techniques and style; design and implementation of programs with multiple modules, using good data structures and good programming style. Prerequisite: CS 101. FALL, SPRING. [3]

CS 212. Discrete Structures. A broad survey of the mathematical tools necessary for an understanding of computer science. Topics covered include an introduction to sets, relations, functions, basic counting techniques, permutations, combinations, graphs, recurrence relations, simple analysis of algorithms, O -notation, Boolean algebra, propositional calculus, and numeric representation. Prerequisite: A course in computer science or two semesters of calculus. FALL, SPRING. [3]

CS 231. Computer Organization. The entire hierarchical structure of computer architecture, beginning at the lowest level with a simple machine model (e.g., a simple von Neumann machine). Processors, process handling, I/O handling, and assembler concepts. Graduate credit not given for computer science majors. Prerequisite: CS 201; corequisite: EECE 116/116L. FALL, SPRING. [3]

CS 240a–240b. Undergraduate Research. Open to qualified majors with consent of instructor and adviser. No more than 6 hours may be counted towards the computer science major. Prerequisite: CS 231. FALL, SPRING. [Variable credit: 1–3 each semester, not to exceed a total of 6]

CS 242. Special Topics in Computer Science. [Variable credit: 1–3]

CS 250. Algorithms. Advanced data structures, systematic study and analysis of important algorithms for searching; sorting; string processing; mathematical, geometrical, and graph algorithms, classes of P and NP, NP-complete and intractable problems. Prerequisite: CS 201 and CS 212. FALL, SPRING. [3]

CS 251. Intermediate Software Design. High quality development and reuse of architectural patterns, design patterns, and software components. Theoretical and practical aspects of developing, documenting, testing, and applying reusable class libraries and object-oriented frameworks using object-oriented and component-based programming languages and tools. Prerequisite: CS 201. FALL, SPRING [3]

CS 252. Theory of Automata, Formal Languages, and Computation. Finite-state machines and regular expressions. Context-free grammars and languages. Pushdown automata. Turing machines. Undecidability. The Chomsky hierarchy. Computational complexity. Prerequisite: CS 212. SPRING. [3]

CS 253. Image Processing. (Also listed as EECE 253) The theory of signals and systems is extended to two dimensions. Coverage includes filtering, 2-DFFTs, edge detection, and image enhancement. Three lectures and one laboratory period. FALL. [4]

CS 255. Introduction to Numerical Mathematics. (Also listed as Math 226) Numerical solution of linear and nonlinear equations, interpolation, and polynomial approximation theory, numerical solution of differential equations, errors and floating point arithmetic. Application of the theory to problems in science, engineering, and economics. Student use of the computer is emphasized. Prerequisite: computer programming and linear algebra. FALL, SPRING. [3]

CS 257. Linear Optimization. (Also listed as Math 288) An introduction to linear programming and its applications. Formulation of linear programs. The simplex method, duality,

complementary slackness, dual simplex method and sensitivity analysis. The ellipsoid method. Interior point methods. Possible additional topics include the primal-dual algorithm, cutting planes, or branch-and-bound. Applications to networks, management, engineering and physical sciences. Prerequisite: linear algebra and computer programming. SPRING. [3]

CS 258. Introduction to Computer Graphics. Featuring 2D rendering and image-based techniques, 2D and 3D transformations, modeling, 3D rendering, graphics pipeline, ray-tracing, and texture-mapping. Prerequisite: Linear Algebra, CS 201, junior standing. FALL. [3]

CS 259. Introduction to Computer Animation. Introduction to the principles and techniques of computer animation. Students work in small groups on the design, modeling, animation, and rendering of a small animation project. Topics include storyboarding, camera control, skeletons, inverse kinematics, splines, keyframing, motion capture, dynamic simulation, particle systems, facial animation, and motion perception. Prerequisite: CS 201, Linear Algebra. SPRING. [3]

CS 260. Artificial Intelligence. Introduction to the principles and programming techniques of artificial intelligence. Strategies for searching, representation of knowledge and automatic deduction, learning, and adaptive systems. Survey of applications. Prerequisite: CS 250 and CS 270 or consent of instructor. FALL. [3]

CS 265. Introduction to Database Management Systems. Logical and physical organization of databases. Data models and query languages, with emphasis on the relational model and its semantics. Concepts of data independence, security, integrity, concurrency. Prerequisite: CS 201. FALL. [3]

CS 269. Project in Artificial Intelligence. Students work in small groups on the specification, design, implementation, and testing of a sizeable AI software project. Projects (e.g., an "intelligent" game player) require that students address a variety of AI subject areas, notably heuristic search, uncertain reasoning, planning, knowledge representation, and learning. Class discussion highlights student progress, elaborates topics under investigation, and identifies other relevant topics (e.g., vision) that the project does not explore in depth. Prerequisite: CS 260 or consent of instructor. SPRING. [3]

CS 270. Programming Languages. General criteria for design, implementation, and evaluation of programming languages. Historical perspective. Syntactic and semantic specification, compilations, and interpretation processes. Comparative studies of data types and data control, procedures and parameters, sequence control, nesting, scope and storage management, run-time representations. Non-standard languages, problem-solving assignments in a laboratory environment. Prerequisite: CS 231. FALL, SPRING. [4]

CS 274. Modeling and Simulation. General theory of modeling and simulation of a variety of systems: physical processes, computer systems, biological systems, and manufacturing processes. Principles of discrete-event, continuous, and hybrid system modeling, simulation algorithms for the different modeling paradigms, methodologies for constructing models of a number of realistic systems, and analysis of system behavior. Computational issues in modeling and analysis of systems. Stochastic simulations. Prerequisite: CS 201, Math 194 or Math 198, Math 216 or Math 218. SPRING. [3]

CS 276. Compiler Construction. Review of programming 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 a high-level language to write a complete compiler. Prerequisite: CS 231. FALL. [3]

CS 278. Principles of Software Engineering. The nature of software. The object-oriented paradigm. Software life-cycle models. Requirements, specification, design, implementation, documentation, and testing of software. Object-oriented analysis and design. Software maintenance. Prerequisite: CS 270 or senior standing in Computer Science or Computer Engineering. FALL. [3]

CS 279. Software Engineering Project. Students work in teams to specify, design, implement, document, and test a nontrivial software project. The use of CASE (Computer-Assisted Software Engineering) tools is stressed. Prerequisite: CS 278. SPRING. [3]

CS 281. Principles of Operating Systems I. Overview of goals of operating systems. Introduction to the resource allocation and control functions of operating systems. Scheduling of processes and processors. Concurrent processes and primitives for their synchronization. Use of parallel processes in designing operating system subsystems. Methods of implementation of parallel processes on conventional computers. Introduction of notions of virtual memory, paging, protection of shared and non-shared information. Structures of files of data in secondary storage. Security issues. Case studies. Prerequisite: CS 231. FALL, SPRING. [3]

CS 282. Principles of Operating Systems II. Projects involving modification of a current operating system. Lectures on memory management policies, including virtual memory. Protection and sharing of information, including general models for implementation of various degrees of sharing. Resource allocation in general, including deadlock detection and prevention strategies. Introduction to operating system performance measurement, for both efficiency and logical correctness. Two hours lecture and one hour laboratory. Prerequisite: CS 281. SPRING. [3]

CS 283. Computer Networks. Computer communications, network architectures, protocol hierarchies, and the open systems interconnection model. Modeling, analysis and specification of protocols. Wide area networks and local area networks including rings, buses, and contention networks. Prerequisite: CS 281. SPRING. [3]

CS 284. Computer Systems Analysis. Techniques for evaluating computer system performance with emphasis upon application. Topics include measurement and instrumentation techniques, benchmarking, simulation techniques, elementary queuing models, data analysis, operation analysis, performance criteria, case studies. Project involving a real computer system. Prerequisite: CS 281. SPRING. [3]

CS 285. Network Security. Principles and practice of network security. Security threats and mechanisms. Cryptography, key management, and message authentication. System security practices and recent research topics. Prerequisite: CS 283. FALL. [3]

CS 291–292. Special Topics. [Variable credit: 1–3 each semester] (Offered on demand)

CS 310. Design and Analysis of Algorithms. Set manipulation techniques, divide-and-conquer methods, the greedy method, dynamic programming, algorithms on graphs, backtracking, branch-and-bound, lower bound theory, NP-hard and NP-complete problems, approximation algorithms. Prerequisite: CS 250. SPRING. [3]

CS 311. Graph Algorithms. Algorithms for dealing with special classes of graphs. Particular emphasis is given to subclasses of perfect graphs and graphs that can be stored in a small amount of space. Interval, chordal, permutation, comparability, and circular-arc graphs; graph decomposition. Prerequisite: CS 310 or Math 275. FALL. [3]

CS 315. Automated Verification. Systems verification and validation, industrial case studies, propositional and predicate logic, syntax and semantics of computational tree and linear time logics, binary decision diagrams, timed automata model and real-time verification, hands on experience with model checking using the SMV, SPIN and UPPAAL tools, and state reduction techniques. Fall. [3]

CS 320. Algorithms for Parallel Computing. Design and analysis of parallel algorithms for sorting, searching, matrix processing, FFT, optimization, and other problems. Existing and proposed parallel architectures, including SIMD machines, MIMD machines, and VLSI systolic arrays. Prerequisite: CS 310 or consent of instructor. [3]

CS 343. High-Performance Computing for Engineers. (Also listed as ME 343) Introduction to high-performance computing. Engineering applications. Focus on high-speed cluster computing. Class project applying high-performance computing to various research topics. SPRING. [3]

CS 350. Artificial Neural Networks. (Also listed as BME 350 and EECE 350) Theory and practice of parallel distributed processing methods using networks of neuron-like computational devices. Neurobiological inspirations, attractor networks, correlational and error-correction learning, regularization, unsupervised learning, reinforcement learning, Bayesian and information theoretic approaches, hardware support, and engineering applications. SPRING. [3]

CS 351. Advanced Animation. Current research issues and problems in computer animation, with special focus on motion capture, dynamic simulation, and key-framing. Cloth, deformable bodies, natural phenomena, geometric algorithms, procedural techniques, facial animation, hair, autonomous characters, flocking, empirical evaluation, and interfaces for animation. Prerequisite: CS 259 or consent of instructor. FALL. [3]

CS 352. Human-Computer Interaction. An overview of human computer interaction and problems of current interest. Topics include: Human factors, GOMS, user interface design and evaluation, interaction modalities, distributed cognition, ubiquitous computing. A project involving design and evaluation will be performed. Prerequisite: consent of instructor. FALL. [3]

CS 357. Advanced Image Processing. (Also listed as EECE 357) Techniques of image processing. Topics include image formation, digitization, linear shift-invariant processing, feature detection, and motion. Prerequisite: Math 175; programming experience. FALL. [3]

CS 358. Computer Vision. (Also listed as EECE 358) The fundamentals of computer vision and techniques for image understanding and high-level image processing. Includes image segmentation, geometric structures, relational structures, motion, matching, inference, and vision systems. Prerequisite: CS 357 or EECE 357. SPRING. [3]

CS 359. Medical Image Registration. Foundations of medical image registration. Mathematical methods and practical applications. Image-to-image registration, image-to-physical registration, applications to image-guided procedures and the most commonly used imaging modalities with an emphasis on tomographic images. FALL. [3]

CS 360. Advanced Artificial Intelligence. Discussion of state-of-the-art and current research issues in heuristic search, knowledge representation, deduction, and reasoning. Related application areas include: planning systems, qualitative reasoning, cognitive models of human memory, user modeling in ICAI, reasoning with uncertainty, knowledge-based system design, and language comprehension. Prerequisite: CS 260 or equivalent. FALL. [3]

CS 362. Machine Learning. An introduction to machine learning principles of artificial intelligence, stressing learning's role in constraining search by augmenting and/or reorganizing memory. Topics include connectionist systems; concept learning from examples; operator, episode, and plan learning; problem-solving architectures that support learning; conceptual clustering; computer models of scientific discovery; explanation-based learning; and analogical reasoning. Psychological as well as computational interests in learning are encouraged. Prerequisite: CS 260, CS 360, or equivalent. SPRING. [3]

CS 364. Intelligent Learning Environments. (Also listed as EECE 355) Theories and concepts from computer science, artificial intelligence, cognitive science, and education that facilitate designing, building, and evaluating computer-based instructional systems. Development and substantiation of the concept, architecture, and implementation of intelligent learning environments. Multimedia and web-based technology in teaching, learning, collaboration, and assessment. Prerequisite: CS 260, CS 360, or equivalent. SPRING. [3]

CS 366. Distributed Artificial Intelligence. Principles and practice of multiple agent systems for distributed artificial intelligence. Game theory, distributed negotiation and decision making, distributed problem solving, cooperation, coalition formation and distributed learning. Prerequisite: CS 260. SPRING. [3]

CS 369. Master's Thesis Research. [0]

CS 375. Discrete-Event Systems: Supervisory Control and Diagnosis. Algebraic structures, automata and formal language theory, process modeling with finite-state automata, supervisory control theory, controllability and supervision, supervisory control under partial observation, modular and hierarchical supervisory control, supervisory control of real-time systems, fault diagnosis of discrete-event systems, and modular diagnosis approaches. SPRING. [3]

CS 376. Foundations of Hybrid and Embedded Systems. Modeling, analysis, and design of hybrid and embedded systems. Heterogeneous modeling and design of embedded systems using formal models of computation, modeling and simulation of hybrid systems, properties of hybrid systems, analysis methods based on abstractions, reachability, and verification of hybrid systems. FALL. [3]

CS 379. Topics in Embedded Software and Systems. Specification and composition of domain-specific modeling languages. Design methodologies for embedded systems. Platforms for embedded system design and implementation. Analysis of embedded systems. SPRING. [3]

CS 381. Advanced Operating Systems Principles. Techniques for formally analyzing various issues in operating systems. Includes process synchronization, interprocess communication, deadlock, naming, memory management, objective capability-models, architectural support, protection, fault tolerance. Prerequisite: CS 281. FALL. [3]

CS 384. Performance Evaluation of Computer Systems. Techniques for computer systems modeling and analysis. Topics covered include analytical modeling with emphasis on queuing network models, efficient computational algorithms for exact and approximate solutions, parameter estimation and prediction, validation techniques, workload characterization, performance optimization, communication and distributed system modeling. Prerequisite: CS 281 or CS 381. SPRING. [3]

CS 385. Advanced Software Engineering. An intensive study of selected areas of software engineering. Topics may include CASE tools, formal methods, generative techniques, aspect-oriented programming, metrics, modeling, reuse, software architecture, testing, and open-source software. Prerequisite: CS 278. FALL. [3]

CS 386. System-Level Fault Diagnosis. An overview of the basic concepts of the theory of fault diagnosis and problems of current interest. Topics include the classical PMC and BGM models of fault diagnosis, hybrid (permanent and intermittent faults) models, diagnostic measures for one-step, sequential, and inexact diagnosis. Emphasis is on algorithmic techniques for solving the diagnosis and diagnosability problems in various models. Prerequisite: CS 381 or consent of instructor. SPRING. [3]

CS 387. Topics in Software Engineering. Topics may include empirical software engineering and open-source software engineering. Prerequisite: CS 278 or consent of instructor. SPRING. [3]

CS 388. Model-Integrated Computing. Model-Integrated Computing addresses the problems of designing, creating, and evolving information systems by providing rich, domain-specific modeling environments including model analysis and model-based program synthesis tools. Students are required to give a class presentation and prepare a project. FALL. [3]

CS 389. Master of Engineering Project.

CS 390. Individual Studies. Offered each term. [1–3]

CS 391–392. Seminar. [1–3 each semester]

CS 395–396. Special Topics. [3–3]

CS 399. Ph.D. Dissertation Research.

Electrical Engineering

CHAIR Daniel M. Fleetwood

ASSOCIATE CHAIR A. B. Bonds

DIRECTOR OF UNDERGRADUATE STUDIES A. B. Bonds

DIRECTOR OF GRADUATE STUDIES Bharat L. Bhuva

PROFESSORS EMERITI Arthur J. Brodersen, James A. Cadzow, George E. Cook, Jimmy L. Davidson, L. Ensign Johnson, Robert T. Nash, Richard G. Shiavi, Francis M. Wells, Edward J. White

PROFESSORS A. B. Bonds, Benoit M. Dawant, Daniel M. Fleetwood, Kenneth F. Galloway, Dennis G. Hall, Weng Poo Kang, Gábor Karsai, Kazuhiko Kawamura, Lloyd W. Massengill, Ronald D. Schrimpf, Janos Sztipanovits, Robert A. Weller

PROFESSOR OF THE PRACTICE Andrew W. Dozier

ASSOCIATE PROFESSORS Bharat L. Bhuva, Richard Alan Peters II, Robert A. Reed, Greg Walker, D. Mitchell Wilkes, James E. Wittig

RESEARCH ASSOCIATE PROFESSORS Michael L. Alles, Theodore Bapty, William T. Holman, Akos Ledeczki, Arthur F. Witulski

ASSISTANT PROFESSORS Zhaohua Ding, William H. Robinson, Sharon M. Weiss

RESEARCH ASSISTANT PROFESSOR Sandeep Neema

ASSISTANT PROFESSOR OF THE PRACTICE Lason L. Watai

✂ THE electrical engineer has been primarily responsible for the information technology revolution that society is experiencing. The development of large-scale integrated circuits has led to the development of computers and networks of ever-increasing capabilities. Computers greatly influence the methods used by engineers for designing and problem solving.

The curricula of the electrical engineering and computer engineering majors are multifaceted. They provide a broad foundation in mathematics, physics, and computer science and a traditional background in circuit analysis and electronics. Several exciting areas of concentration are available, including microelectronics, computer systems, robotics and control systems, and signal processing. Double majors may be arranged with some programs, including biomedical engineering and mathematics. Students receive an education that prepares them for diverse careers in industry and government and for postgraduate education.

Undergraduate Honors Program. With faculty approval, junior and senior students may be accepted into the Honors Program. To achieve honors status, the student must: