Computer Science Undergraduate Handbook

University of Utah
School of Computing
50 S Central Campus Dr RM 3190
Salt Lake City, UT 84112-9205

(801) 581-8224 (voice)
(801) 581-5843 (fax)
info@cs.utah.edu
http://www.cs.utah.edu

2004–2005 Academic Year
June 2004

The School of Computing offers a Bachelor of Science degree in Computer Science. The undergraduate program begins with a set of three courses that give students a solid background in programming and discrete mathematics while exposing them to the breadth of issues that arise in computer science. Students then take five core courses in software engineering, computer organization, algorithms and data structures, software systems, and theory. They build on this background by choosing seven electives from the breadth of the School's course offerings (which includes advanced courses in theoretical computer science, scientific computing, artificial intelligence, databases, operating systems, computer networks, programming languages, graphics, computer architecture, and digital design). Each student's undergraduate program is capped with a senior project. Along with an in-depth study of computing, the curriculum encompasses a general education in mathematics, science, and the humanities.

The School also offers an honors version of the Bachelor of Science degree for students who are contemplating a career in research, a combination BS/MS degree for students who wish to complete both the Bachelor and Master of Science degrees in a total of five years, and a minor in Computer Science for students who want to use computers in another field. In addition, selected service courses are offered to provide an introduction to the use of computers as tools for students of many backgrounds and interests.

A Bachelor of Science in Computer Engineering is jointly offered by the School of Computing and the Department of Electrical and Computer Engineering. Information about that program is available in a separate handbook or from the web page referenced below.

(The latest version of this handbook is available online at `http://www.cs.utah.edu/dept/handbooks`.)

Faculty

Professors	Martin Berzins, Ph.D.	Elaine Cohen, Ph.D.
	Alan L. Davis, Ph.D.	Ganesh Gopalakrishnan, Ph.D.
	Thomas C. Henderson, Ph.D.	Lee A. Hollaar, Ph.D.
	John M. Hollerbach, Ph.D.	Christopher R. Johnson, Ph.D.
	Robert Kessler, Ph.D.	Gary E. Lindstrom, Ph.D.
	Richard F. Riesenfeld, Ph.D.	Kris Sikorski, Ph.D.
	Frank Stenger, Ph.D.	William B. Thompson, Ph.D.
Clinical Professors	David Hanscom, Ph.D.	Joseph L. Zachary, Ph.D.
Research Professor	Stephen Jacobsen, Ph.D.	Jay Lepreau
Emeritus Professors	Robert R. Johnson, Ph.D.	Kent Smith, Ph.D.
Associate Professors	Erik Brunvand, Ph.D.	John Carter, Ph.D.
	Charles Hansen, Ph.D.	Ellen M. Riloff, Ph.D.
	Peter Shirley, Ph.D.	Claudio Silva, Ph.D.
	Ross Whitaker, Ph.D.	
Clinical Associate Professor	Art Lee, Ph.D.	
Research Associate Professors	Sam Drake, Sc.D.	
Assistant Professors	Rajeev Balusubramonium, Ph.D.	Matthew Flatt, Ph.D.
	Sneha Kasera, Ph.D.	Mike Kirby, Ph.D.
	Emil Praun, Ph.D.	John Regehr, Ph.D.
	Konrad Slind, Ph.D.	Cynthia Thompson, Ph.D.
Research Assistant Professors	Steven Parker, Ph.D.	
Adjunct Associate Professors	Robert McDermott, Ph.D.	Chris Myers, Ph.D.
	Wilson Hsieh, Ph.D.	
Adjunct Assistant Professor	Sally Creem-Regehr, Ph.D.	Oleg Portniaguine, Ph.D.

Administration

Christopher Johnson	Director	581-6456
Martin Berzins	Associate Director	585-1545
Al Davis	Associate Director	581-3991
Dave Hanscom	Director, Undergraduate Studies	581-7023
Erik Brunvand	Director, Computer Engineering	581-4345
Joe Zachary	Director, Educational Programs	581-7079
Peter Shirley	Director, Graduate Studies	585-1883
Kris Sikorski	Director, Computational Engineering and Science	581-8579

Administrative Staff

Ann Torrence	Development, Industry Relations	581-7631
Phil Willden	Office Manager	581-3950
Sara Mathis	Administrative Secretary	581-8226
Chris Coleman	Publications, Outreach, Alumni Services	581-8580
Erin Bergey	Receptionist	581-8224

Academic Advising

Sandy Hiskey	Undergraduate Academic Counselor	581-8224
Karen Feinauer	Graduate Counselor	581-8224
Emily Wardle	Graduate Admissions Coordinator	581-8224

Technical Staff and Services

Corey Hatch	Digital Systems Laboratory Manager	581-7845
Monica Heaton	Administrative Officer	581-9371
Chad Lake	Computing Facility Director	585-7614
Mark Gully	Undergraduate Lab Manager	585-9047

A complete staff listing can be found at http://www.cs.utah.edu/people.html.

Contents

1 The Computer Science Major and Minor

The School of Computing offers a Bachelor of Science in Computer Science. This is a software-oriented degree whose requirements include 17 computer science courses. A student must be admitted as a computer science major by the School in order to take upper-division courses and pursue the computer science degree. The School also offers an honors version of the Bachelor of Science, a 5-year combination Bachelors/Masters degree, as well as a minor in computer science consisting of six computer science courses.

1.1 Becoming a Computer Science Major

Any student can become a computer science pre-major by informing the University Registrar or the School of Computing Academic Counselor. It is advisable to do this early to ensure receiving information about the major and staying advised of any changes that may be made in degree requirements. Declaration of a major will also enable participation in activities associated with the degree program such as the Undergraduate Student Advisory Committee.

In order to become a full major, a student must complete the courses required of pre-majors and apply for full major status. An application should be filled out at the School of Computing office during the semester when he or she expects to complete these requirements. A student may not preregister for any upper division classes in computer science without first being admitted as a full major or a minor. Applications for admission are reviewed at the end of each semester.

To be admitted to full major status, a student must have:

1. An average grade of at least 2.7 and a minimum grade of C− in the following classes or their equivalents. *None of these classes may be taken on a credit/no-credit basis.*
   - Computer Science 2010 and 2020.
   - Mathematics 1210 and 1220.
2. A grade of CR in CPSC 1010 (a credit/no-credit class).
3. A cumulative University of Utah grade point average of 2.3 or higher.

If credit is granted for any of the above classes based on advanced placement test scores or courses taken at other schools, grades are assigned for use in the calculation. Check with the Undergraduate Academic Counselor for details.

1.2 Undergraduate Advising

The School of Computing has an Undergraduate Academic Counselor who is available to answer questions regarding registration for computer science classes, transfer of credits, degree requirements, recent School actions, etc. (Sandy Hiskey, 3190 MEB, `shiskey@cs.utah.edu`.) The School also has a faculty advisor, who can answer questions about any of the above, as well as more technical issues, such as career decisions. Students are welcome to meet with the Academic Counselor or with the faculty advisor whenever necessary to discuss schedule plans, graduation requirements, or current problems. The responsibility for arranging an appointment is left to the student. Call the School of Computing receptionist (MEB 3190, 581-8224) to set up a time to meet with either the Academic Counselor or faculty advisor.

1.3 Requirements for the Bachelor of Science Degree

The computer science degree can be completed in four full-time years of study if the student is capable of completing the above computer science and calculus courses during the freshman year. If a student must instead take preparatory classes as a freshman, more than four years may be required to earn a degree. In any event, it is important to take the required pre-major classes early to allow advancement to full major status as soon as possible.

1. **General Education:** The General Education requirements are described in the University of Utah General Catalog. The requirements for computer science majors are more specific.

   (a) The University writing requirement is required for computer science pre-majors.

   (b) The quantitative reasoning requirement is satisfied by Math 1210/1220 or 1210/1250, which are required for computer science pre-majors.

   (c) Students must take two intellectual explorations courses in each of fine arts, humanities, and social sciences. (The two-course requirement in physical and life sciences is automatically satisfied by classes required for the major.) *These six courses must include two upper division courses.*

   A General Education Approval form can be obtained from the Academic Counselor. Students must complete this form and receive approval for their programs.

   (d) The American Institutions requirement can be satisfied by taking one of Economics 1740, History 1700, Honors 2212, or Political Science 1100.

   Incoming engineering students should consider the Engineering LEAP (E-LEAP) program organized by the Office of Undergraduate Studies. It is a year-long cohort program set up to allow students to take several classes together during their first year on campus. Two of these classes are seminar courses that satisfy University General Education and Diversity requirements. The program also includes sections of Writing, Calculus, and Physics. For more information on E-LEAP, contact the College of Engineering Advisor at 585-7769.

   Computer Science students should also consider CPSC 1050, Computers in Society, as one of their Social/Behavioral Science General Education classes. The course focuses on the social issues that surround the increasingly pervasive roles that computers play in society.

2. **University Graduation Requirements:** The University graduation requirements for the Bachelor of Science degree are described in the University of Utah General Catalog.

   (a) The communication/writing requirement is satisfied by either Writing 3200 or 3400, one of which is required for computer science majors.

   (b) The quantitatively intensive course requirement is satisfied by CPSC 3510 and 3810, which are required for computer science majors.

   (c) The diversity requirement can be satisfied by taking a course from the approved list as part of the intellectual explorations requirement.

   (d) Students must complete a minimum of 122 semester hours of course work. At least 40 of the 122 hours must be upper division classes. (Upper division classes are numbered 3000 or above. Credits from two-year colleges will not count toward University upper division hours.) At least 30 of the total credit hours and 20 of the last 30 hours must be taken at the University.

3. **Writing:** Two classes in writing are required:

   (a) Either Writing 2010 or ESL 1060 (for students who speak English as a second language).

   (b) Either Writing 3200 or 3400. Writing 3400 is recommended for most students; Writing 3200 is an option for transfer students.

4. **Math, Science, and Engineering:** Seven classes in math, science, and/or engineering are required.

   (a) Mathematics 1210/1220 or 1210/1250 (Calculus I and II)

   (b) Physics 2210 (Physics I)

   (c) Four additional courses, *each of which must be at least three semester hours*, chosen from among Physics 2220 (Physics II), Biology 1000 (General Biology), Chemistry 1210 (General Chemistry), or any class (other than a computer science class) from the Colleges of Engineering, Mines, or Science that requires a full year of calculus as a prerequisite or corequisite. Students should take the prerequisites of computer science electives into consideration when planning how to satisfy this requirement.

5. **Computer Science:** A minimum of 17 computer science classes must be taken. Figure 1.1 gives an example four-year degree program leading to a Bachelor's Degree in computer science. Figure 1.2 summarizes the prerequisites for computer science courses.

   (a) Required. The following classes must be taken:

CPSC 1010	Introduction to Unix
CPSC 2010	Introduction to Computer Science I
CPSC 2020	Introduction to Computer Science II
CPSC 2100	Discrete Structures
CPSC 3400	Computer Systems
CPSC 3500	Software Practice
CPSC 3510	Algorithms and Data Structures
CPSC 3810	Computer Organization

   (b) Theory restricted elective. Students must take one of the following.

CPSC 3100	Models of Computation
CPSC 3200	Scientific Computing

   (c) Electives. Seven additional Computer Science classes numbered 3000 or higher, totaling at least 21 semester hours, must be taken. CPSC 5010/20 and seminars may not be counted. Only one of CPSC 5600/5605 and only one independent study class (with special permission) may be counted.

   (d) Capstone Requirement. One of the following must be completed.

CPSC 4500	Software Engineering Lab
CPSC 4970[†]	Bachelor's Thesis

   [†] *Students choosing the thesis option must get special permission from the faculty.* Such approval is usually obtained by finding a faculty advisor in the Spring of the Junior year. This option is intended for students who are considering graduate school. *The Bachelor's Thesis can also be used toward a portion of the thesis requirement for a BS/MS program in Computer Science.* For more information about the joint BS/MS program, please see the director of the BS/MS program in the School of Computing.

   (e) Duplication of credit. No single class may be counted toward more than one of the requirements listed above.

6. **Continuing Performance:** *All computer science, mathematics, science, engineering, and writing courses taken to satisfy the requirements listed above must be passed with a grade of C− or better* (except for CPSC 1010, in which a grade of CR is required). A student may repeat such courses only one time.

   To remain in good standing and graduate, a student must maintain a cumulative grade point average at the University of 2.3 or higher, and also maintain a grade point average (GPA) of 2.3 in computer science classes taken at the University. Students whose GPA in either of these categories falls below 2.3 are notified that they are on probation and will be given conditions for a return to good standing. Normally, these conditions must be satisfied during the next two semesters, excluding summers. Students failing to meet their probationary conditions are dropped from the rolls of the major.

	Fall		*Spring*	
Freshman	CPSC 1010†	(0.5)	CPSC 2020†	(4)
	CPSC 2010†	(4)	CPSC 2100	(3)
	Math 1210†	(4)	Math 1220†	(4)
	Writing 2010	(3)	Gen Ed	(3)
	Gen Ed	(3)		
		(14.5)		(14)
Sophomore	CPSC 3500	(4)	CPSC 3510	(4)
	CPSC 3810	(4)	CS elective	(3)
	Math/sci elective	(4)	Physics 2210	(4)
	Writing 3400	(3)	Gen Ed	(3)
			Free elective	(2)
		(15)		(16)
Junior	CPSC 3400	(4)	CS elective	(3)
	CS theory elective	(3)	CS elective	(3)
	Math/sci elective	(4)	Math/sci elective	(3)
	Gen Ed	(3)	Gen Ed	(3)
	Free elective	(2)	Free elective	(4)
		(16)		(16)
Senior	CS elective	(3)	CPSC 4500	(3)
	CS elective	(3)	CS elective	(3)
	CS elective	(3)	Math/sci elective	(3)
	Gen Ed	(3)	Gen Ed	(3)
	Free elective	(3)	Free elective	(4)
		(15)		(16)

This table gives an eight-semester example program leading to a B.S. in Computer Science. It is meant only as a guide, since the scheduling of electives and General Education classes depends upon which ones are selected. This schedule assumes adequate high school preparation in mathematics; it is not advisable to take Physics 2210 without some previous training in calculus. Note that Math 1210 and Computer Science 2010 must be taken during the fall semester in order to complete the required pre-major classes during the first year, unless the student has advanced placement credit. (†Class required of pre-majors.)

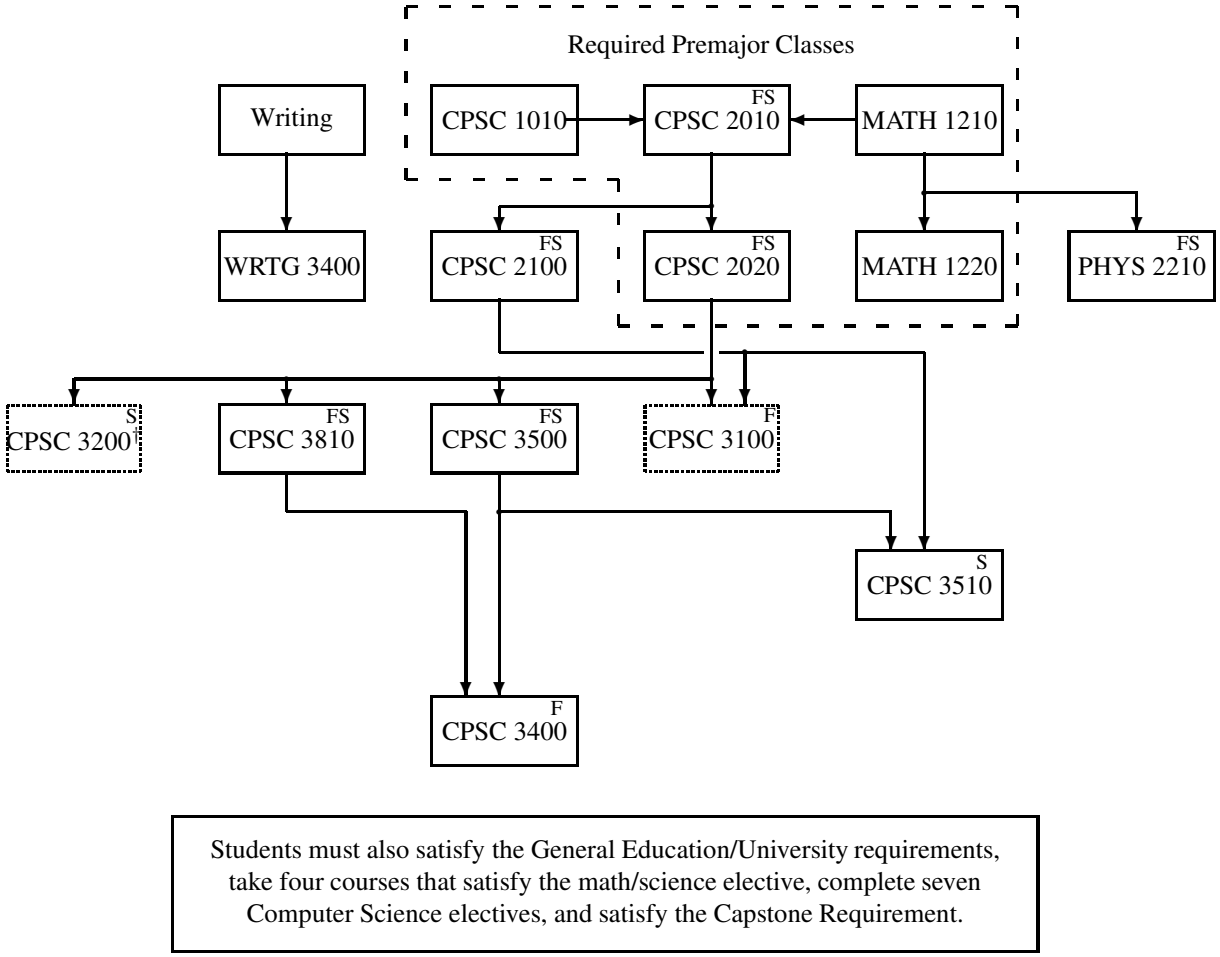
Figure 1.1: Example Computer Science Degree Program

All students admitted as full majors are placed on probationary status. If a student's GPA in either of the above categories is below 2.3 after the first year during which they take upper level computer science classes, the student is dropped from the rolls of the major.

Students are expected to complete all requirements for their degree within four years of acceptance to full major status. Students not making satisfactory progress toward their degrees may be dropped from the rolls and declared inactive. The determination that a student is not making satisfactory progress is made in one of two ways. Either (1) the student has not completed a computer science course for a period of one year, or (2) there is no reasonable way in which the student can complete all degree requirements by the end of the required period of time.

In order to be reinstated from inactive status or from being dropped due to low GPA, students must petition the Computer Science Undergraduate Committee. Reinstated students proceed under the latest graduation requirements.

If personal circumstances prevent completion of all degree requirements within four years of acceptance as a full major, a student may request an extension of a specific duration and submit a revised schedule of completion.



This graph illustrates the order in which classes must be taken to satisfy prerequisite and corequisite requirements in Computer Science. Prerequisites are connected bottom-to-top; corequisites are connected side-to-side. One of the two courses contained in dashed boxes must be taken. Where not otherwise indicated, courses are offered during both semesters as well as the summer. (†CPSC 3200 has Math 2250 as a prerequisite; CPSC 4500 can be taken only by students who are on track to graduate before the next offering of CPSC 4500.)

Figure 1.2: Computer Science Prerequisites

1.4 Computer Science Honors Degree

The School offers an Honors Bachelor's Degree in Computer Science. This program is intended primarily for students thinking of a career in research, but any student can apply. Students who wish to be admitted into the honors track should apply after taking CPSC 2020. This application should consist of a short statement of career goals, along with any supplementary material the student deems relevent. This supplementary material is strictly optional.

Students admitted to the Departmental Honors track will have probationary status until after the completion of CPSC 4005.

The degree requirements for the honors degree are the same as for the conventional Computer Science degree with the following exceptions:

1. CPSC 4005 is required.
2. Either Honors 2211 or Honors 3200 is required (and can be counted in place of one of the required writing courses).
3. At least one Computer Science course at the 6000-level or above must be taken.

4. In addition, at least four courses must be taken that either have an honors designation or are Computer Science classes numbered 6000 or above. At least three of these courses should be offered through the University Honors Curriculum (i.e., have course number Honors-XXXX).

5. A Bachelor's honors thesis is required.

The upper level CPSC classes taken to satisfy the above requirements would be counted toward the Computer Science elective requirement.

At graduation, the University requires that honors students have a GPA of at least 3.5 in the major and at least 3.4 overall.

1.5 Computer Science as a Minor

The School of Computing offers a minor for students who desire to gain sufficient background to use and program computers in another field.

In order to be admitted as a computer science minor, a student must have a declared major in another department and be making progress in that major. The admission process is similar to that for majors and is carried out at the same times. Students are admitted to the minor if their average grade in Math 1210, CPSC 2010, and CPSC 2020 is 2.7 or higher.

The minor consists of a minimum of 18.5 semester hours of computer science classes.

- Required. The following classes must be taken.

CPSC 1010	Introduction to Unix
CPSC 2010/2020	Introduction to Computer Science
CPSC 2100	Discrete Structures
CPSC 3500	Software Practice

- Elective. Students must take at least one additional CPSC class at the 3000 level.

Computer science minors are guaranteed admission into only the upper division classes that comprise their minor.

1.6 Students Pursuing a Second Degree or a Double Major

Some students may wish to earn a degree in computer science as their second B.S. This is possible as long as the requirements for both degrees are met. In some cases, fewer additional class hours are needed because of overlaps in the two degrees. This is especially true of students whose other degree is in computer engineering, electrical engineering, or mathematics, where upper level classes may serve as computer science electives. Students pursuing a double major must notify the Academic Counselor.

1.7 Undergraduate Financial Assistance

The School of Computing has several financial assistance awards available. Applications are available on the web[1]. These scholarships are awarded based upon academic performance, rather than financial need. Most are awarded to Computer Science and Computer Engineering full majors or those students who will become full majors during the following academic year. Applications for these scholarships must be submitted to the School's office by March 15:

[1] http://www.cs.utah.edu/info/forms/scholarship_form.pdf

Special Departmental Scholarships. These awards are available to students majoring in Computer Science and Computer Engineering who are residents of the state of Utah. They are for resident tuition for two semesters. To be eligible, students must take at least 12 credit hours per semester.

School of Computing Scholarships. A few awards are available each year to Computer Science and Computer Engineering majors. These are unrestricted cash awards of $500 to $2,500, made possible in part by generous donations from the School of Computing faculty and the Eccles Foundation. To be eligible, students must take at least 9 credit hours per semester. Use the same application as for the Continuing Student Scholarship.

College of Engineering Scholarships. These scholarships are awarded to students majoring in Computer Science and Computer Engineering. These are unrestricted cash awards of $500 to $2000. To be eligible, students must take at least 12 credit hours per semester. Use the same application as for the Continuing Student Scholarship.

Students may also apply for financial aid from the College of Engineering, which each year awards about 25 Josephine Beam Educational Scholarships. These scholarships are worth approximately $500 and are based on need. Obtain an application from the Office of the Dean of Engineering in Kennecott 214. Applications must be submitted by February 15.

1.8 Employment Opportunities

The University Office of Career Services offers an internship program which allows qualified students to work in their fields of interest for all or part of their junior and/or senior years. This can be done on a full or part time basis, either in Salt Lake City or elsewhere. *Students are paid for their work, but no credit is granted.* The benefits of such experience include exposure to ideas which could help with career decisions, making contacts which may be useful in the future, and valuable experience in an area that is pertinent to current studies. Corporations participating include IBM, 3M, Hewlett Packard, Microsoft, Evans & Sutherland, Novell, Intel, and Unisys. Many of our majors take advantage of this valuable opportunity.

The School employs a number of junior and senior students as computer operators and as teaching assistants. These jobs involve no more than 20 hours of work per week at an appropriate hourly wage. Appointments are made each semester based on student applications, which should be submitted prior to the start of each term. These applications are available on the web[2]. In addition, general inquiries are received periodically from local industry and from University research groups for students who are interested in working part or full time. These are emailed to all Computer Science and Computer Engineering majors. More information may be obtained from the Academic Counselor.

Students seeking employment upon graduation should contact the University Office of Career Services in order to be included on a list supplied to employers. Students not planning to work towards an advanced degree should register with Career Services at the beginning of their senior year, since most companies begin interviewing in the fall semester.

1.9 Student Participation in School Affairs

Opportunities for students to develop their organizational and leadership abilities are available through participation in the Undergraduate Student Advisory Committee (UgSAC), which plays an active role in the School by coordinating the following:

1. Course and faculty teaching evaluations.
2. Representation (one student) at faculty meetings.
3. Announcements to all declared pre-majors and majors.
4. Representation on the College Student Advisory Committee.
5. Organization of Engineering Week activities.

[2] http://www.cs.utah.edu/dept/webforms/current/ugrad-TA.html

6. Organization of lunch meetings for pre-majors and majors.
7. Organization of university and high school programming contests.
8. Feedback on issues affecting students, such as scheduling, curriculum changes, and graduation requirements.
9. GRE preparation classes.

Anyone interested in joining this organizations should contact UgSAC at `ugsac@cs.utah.edu`. Participation, suggestions, and criticisms are solicited.

2 Computing Facilities

The School of Computing provides state of the art computing facilities for both instructional and research use. Both facilities share a common network infrastructure that is based on a Gigabit Ethernet and that provides desktop connections at speeds ranging from Fast Ethernet (100 Mbs) up to Gigabit Ethernet where necessary. The School's network attaches via an OC-12 connection directly to the campus OC-48 ATM mesh, which in turn routes traffic to Abilene (Internet 2), vBNS, and the Internet.

In addition to the shared network infrastructure, the core School of Computing facility supplies many centralized services, including shared disk space (4 Terrabytes), time, web/cgi, ftp, firewall, backups, printing resources, and email. Most services run on Linux-based hosts, ranging from Ultra 10's to an Enterprise 5000. Several large-scale Solaris and Linux machines are also made available for general use.

The instructional computing facility includes almost 200 Unix, Linux, and Windows-based machines. Most of these machines are organized into three laboratories and the remainder are situated in graduate student offices. The Undergraduate Lab in EMCB 210 includes more than 100 Athlon based PC's; roughly 85 of those run Windows XP and the rest are Linux systems. The electronic classroom in MEB 3225 contains 30 Pentium III-based PCs arranged into a classroom configuration. The Student Project Lab (MEB 3105) contains 5 stations with Pentium IV processors and overhead projectors where groups of students can work in groups on software development projects. The CES/Grad Lab in MEB 3161 contains 13 Linux boxes with Athlon processors. Wireless access is available to department machines throughout the School of Computing.

Students in the School of Computing also have access to the College of Engineering Workstation Laboratory, which consists of five servers, more than 100 Sun workstations, and approximately 10 Linux workstations. Wireless access is available for these machines in the classroom building.

The research computing facility is a heterogeneous mix of over 300 machines, including SGI, HP, Sun and Intel-based hardware. The research computing facility includes major laboratories devoted to computer-aided design and graphics, computer systems, asynchronous digital systems and VLSI, robotics and vision, scientific computing and imaging, and information retrieval and natural language processing. These research laboratories contain a wide array of specialized equipment, including

- an SGI Origin 3800 (32 processors);
- an SGI Origin 2000 Reality Monster (96 processors, 8 IR heads);

- a 200-node network testbed and emulation facility;
- an SGI Power Onyx (14 processors, 2 RE2 heads);
- a multi-source nonlinear video editing environment;
- a real-time signal processing lab;
- an image analysis lab;
- equipment for various types of custom hardware design;
- a Sarcos Dextrous Arm, Utah/MIT Dextrous Hand, and PUMA 560 robots; and
- a Sarcos Treadport locomotion interface, several SensAble Phantom haptic interfaces, Fakespace Responsive Workbench, nVision Datavisor HiRes, and a variety of position trackers.

The College of Engineering operates a research-scale integrated circuit (IC) fabrication facility that is used extensively by the School of Computing. Equipment for testing and debugging both internally and externally fabricated circuits is housed in an integrated circuit testing facility that contains state-of-the-art HP, Tektronix and Micromanipulator automated IC testing equipment.

3 Computer Science Courses

The number and title of each course is followed by the number of semester hours it carries, the semester(s) during which it is taught (F=fall, S=spring, U=summer), its prerequisites, its corequisites, and any courses with which it is cross-listed.

Where a course has both a 5000- and 6000-level number, the 5000-level version is intended for undergraduates, and the 6000-level version is for honors and graduate students. The two versions of the class will meet together, but extra work will be expected of honors and graduate students.

Courses that have only 6000-level numbers may be taken by graduate and advanced undergraduate students.

Current class schedules and registration information[1] are available on line.

1000 Engineering Computing (3, FS) Coreq.: CPSC 1010, MATH 1210

Meets with CPSC 1001. Introduction to programming principles and engineering problem solving via computational means using MATLAB (during the first half of the semester) and C (during the second half of the semester). Decomposition of programs into data representations, functions, and control structures. Clean programming practices are emphasized. The MATLAB portion of the course focuses on the implementation of physically-based models, data visualization via plotting, and selected numerical techniques. The C portion of the course introduces basic syntax and special features of the language for engineering implementations.

1001 Engineering Computing using MATLAB (1.5, FS) Coreq.: CPSC 1010, MATH 1210

Meets with CPSC 1000. Introduction to programming principles and engineering problem solving via computational means using MATLAB. Decomposition of programs into data representations, functions, and control structures. Focus on the implementation of physically-based models, data visualization via plotting, and selected numerical techniques. Clean programming practices are emphasized. (This is a half-semester course that meets with CP SC 1000.)

1010 Introduction to Unix (0.5, FSU)

An introduction to the Unix workstations used in the College of Engineering CADE Lab. Topics include the X Windows system, Unix shell commands, file system issues, text editing with Emacs, accessing the World Wide Web with Netscape, and electronic mail. Self-paced course using online teaching aids.

[1] http://www.acs.utah.edu/prod/bin/student

1020 Introduction to Programming in C++ (3, U)

An introduction to essential programming concepts using C++. Laboratory practice.

1021 Introduction to Programming in Java (3, FU)

An introduction to essential programming concepts using Java. Laboratory practice emphasizes object-oriented techniques and web-based application design.

1040 Creating Interactive Web Content (3, FS)

Introduction to the essentials of web page design and object-oriented programming through the use of HTML and JavaScript to create interactive web pages. It is appropriate for any student who is comfortable using a computer to write a paper and browse the Web. This is a 100% online course that can be completed on any computer equipped with a recent version of Netscape Communicator or Internet Explorer.

1050 Computers in Society (3, FS) Social/Behavioral Science Exploration Course

Survey of the social issues that surround the increasingly pervasive roles that computers play in society. Topics include privacy of personal information, encryption and interception of communications, risks posed by unreliable computer systems, freedom of speech in cyberspace, intellectual property as it relates to downloadable media, computer-based crime, and computers in the workplace. Case studies will focus on computer-related issues of current public interest. No background in computer technology is required or assumed beyond the ability to use a computer to send e-mail, browse the web, and write papers.

1950 Independent Study (1 to 4)

1960 Special Topics (1 to 4)

2000 Introduction to Programming in C (4, F) Coreq.: CPSC 1010

Introduction to essential programming concepts using C. Decomposition of programs into functional units; control structures; fundamental data structures of C; recursion; dynamic memory management; low-level programming. Some exposure to C++. Laboratory practice. (Intended for non-CS majors).

2010 Introduction to Computer Science I (4, FS) Coreq.: MATH 1210, CPSC 1010

The first course required for students intending to major in computer science and computer engineering. Introduction to the engineering and mathematical skills required to effectively program computers, and to the range of issues confronted by computer scientists. Roles of procedural and data abstraction in decomposing programs into manageable pieces. Introduction to object-oriented programming. Extensive programming exercises that involve the application of elementary software engineering techniques.

2020 Introduction to Computer Science II (4, FS) Prereq.: CPSC 2010

The second course required for students intending to major in computer science and computer engineering. Introduction to the problem of engineering computational efficiency into programs. Classical algorithms (including sorting, searching, and graph traversal) and data structures (including stacks, queues, linked lists, trees, hash tables, and graphs). Analysis of program space and time requirements. Extensive programming exercises that require the application of elementary techniques from software engineering.

2100 Discrete Structures (3, FS) Prereq.: CPSC 2010

Introduction to propositional logic, predicate logic, formal logical arguments, finite sets, functions, relations, inductive proofs, recurrence relations, graphs, and their applications to Computer Science.

2950 Independent Study (1–4)

2960 Special Topics (1–4)

3100 Models of Computation (3, F) Quantitatively Intensive B.S. Course. Prereq.: CPSC 2020, CPSC 2100

Models of sequential computation, including finite-state automata, push-down automata, and Turing machines.

3200 Scientific Computation (3, S) Prereq.: CPSC 2020, MATH 2250

Scientific computation relevant to computer science and engineering; floating-point arithmetic, systems of linear equations (direct and iterative techniques), nonlinear equations (univariate and multivariate), interpolation and differentiation (divided differences), integration (mechanical and Gaussian quadratures, optimal quadratures), approximation by spline functions (natural splines and B-splines, optimality of splines).

3400 Computer Systems (4, F) Prereq.: CPSC 3500, CPSC 3810

Introduction to computer systems from a programmer's point of view. Machine level representations of programs, optimizing program performance, memory hierarchy, linking, exceptional control flow, measuring program performance, virtual memory, concurrent programming with threads, network programming.

3500 Software Practice (4, FS) Prereq.: CPSC 2020

Meets with CPSC 5010. Practical exposure to the process of creating large software systems, including requirements specifications, design, implementation, testing, and maintenance. Emphasis on software process, software tools (debuggers, profilers, source code repositories, test harnesses), software engineering techniques (time management, code and documentation standards, source code management, object-oriented analysis and design), and team development practice. Much of the work will be in groups and will involve modifying preexisting software systems.

3510 Advanced Algorithms and Data Structures (4, S) Quantitatively Intensive B.S. Course. Prereq.: CPSC 2100, CPSC 3500

Meets with CPSC 5020. Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CPSC 2020. Balanced trees, heaps, hash tables, string matching, graph algorithms, external sorting and searching. Dynamic programming, exhaustive search. Space and time complexity, derivation and solution of recurrence relations, complexity hierarchies, reducibility, NP completeness. Laboratory practice.

3520 Programming Language Concepts (3, F) Prereq.: CPSC 3500

Ideas behind the design and implementation of programming languages. Syntactic description; scope and lifetime of variables; runtime stack organization; parsing and abstract syntax; semantic issues; type systems; programming paradigms; interpreters and compilers.

3700 Fundamentals of Digital System Design (4, S) Quantitatively Intensive B.S. Course. Cross-listed as ECE 3700. Prereq.: CPSC 2010 or CPSC 2000, PHYCS 2220

Techniques for reasoning about, designing, minimizing and implementing digital circuits and systems. Combinatorial (logic and arithmetic) and sequential circuits are covered in detail leading up to the design of complete small digital systems using finite state machine controllers. Use of computer-aided tools for design, minimization, and simulation of circuits. Laboratory is included involving circuit implementation with MSI, LSI, and field programmable gate arrays.

3710 Computer Design Laboratory (3, F) Cross-listed as ECE 3710. Prereq.: CPSC/ECE 3700, CPSC/ECE 3810

Working in teams, students employ the concepts of digital logic design and computer organization to design, implement, and test a computing system. Interface I/O devices and develop associated software/firmware. Extensive use of CAD and software tools.

3720 Embedded System Design (4, S) Cross-listed as ECE 3720. Prereq.: CPSC/ECE 3700, CPSC/ECE 3810

Introduction to issues in embedded system design using microcontrollers. Topics include: microcontroller architecture, memory interfacing, serial and parallel I/O interfacing, analog interfacing, interrupt synchronization, and embedded software.

3810 Computer Organization (4, FS) Quantitatively Intensive B.S. Course. Cross-listed as ECE 3810. Prereq.: CPSC 2020 or CPSC 2000

An in-depth study of computer architecture and design, including topics such as RISC and CISC instruction set architectures, CPU organizations, pipelining, memory systems, input/output, and parallel machines. Emphasis is placed on performance measures and compilation issues.

3950 Independent Study (1–4)

3960 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2004–05 academic year. Contact the faculty member in charge for details. Check the on-line schedule for a more current listing of offerings.

- **CPSC 3960 Introduction to Computer Networks** (3,F). Prof. Hollaar.
- **CPSC 3960 Team Software Engineering** (3,S). Prof. Kessler.

3991 Computer Engineering Junior Seminar (0.5, F) Cross-listed as ECE 3991. Prereq.: CE major status

Presentation from faculty and industry representatives to discuss trends in computer engineering, professionalism, ethics, the impact of engineering in global and societal contexts, lifelong learning, and contemporary issues.

3992 Computer Engineering Pre-Thesis/Pre-Project (0.5, S) Cross-listed as ECE 3992. Prereq.: CPSC/ECE 3991, CE major status

Fundamentals of project planning (scoping, group selection, risk assessment, scheduling, backup planning, strategy, etc.) are covered in the first half of the course. The second half involves student presentations and critique of proposals in progress. The final result of the course will be an approved project or thesis proposal.

4005 Honors Research Practice (3, F) Prereq.: CPSC 3500 and Honors status

Techniques for identifying a Computer Science research problem, literature review, research execution, and preparation for publication.

4010 Teaching Introductory Computer Science (1, FS) Prereq.: Permission of instructor

Issues confronted by undergraduate teaching assistants in introductory computer science courses, including leading lab sections, conducting office hours, grading assignments, communicating with students. Each student must currently be an undergraduate teaching assistant in the School of Computing. May be taken for credit up to three times.

4500 Software Engineering Laboratory (3, S) Prereq.: CPSC 3510, senior standing in Computer Science

Development of significant software systems by small student groups, with emphasis on applying sound, disciplined software engineering practice.

4540 Web Software Architecture (3, S) Prereq.: CPSC 3510

Software architectures, programming models, and programming environments pertinent to developing web applications. Topics include client-server model, multi-tier software architecture, client-side scripting (JavaScript), server-side programming (Servlets and JavaServer Pages), component reuse (JavaBeans), database connectivity (JDBC), and web servers.

4550 Simulation (3, F) Prereq.: CPSC 3510

Basic simulation modeling, modeling complex systems, basic probability and statistics for simulation, building valid simulations, random numbers, and output data analysis. Both discrete event and continuous simulation may be covered.

4710 Computer Engineering Senior Project (3, F) Cross-listed as ECE 4710. Prereq.: CPSC/ECE 3710, CPSC/ECE 3720, CPSC/ECE 3992

This is the capstone project course for Computer Engineering majors who do not choose to do a thesis. Projects are done in groups and are of the student's choosing. Classroom sessions are devoted to improving presentation skills and serve as peer reviews of the ideas and work done to date. Multiple in-progress oral presentations are required as is a final written project report and a final oral presentation.

4950 Independent Study (1–4)

4960–4964 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2004–05 academic year. Contact the faculty member in charge for details. Check the on-line schedule for a more current listing of offerings.

- **CPSC 4960 Programming Challenges** (1,F). Profs. Silva and Praum
- **CPSC 4960 Programming Challenges** (1,S). Profs. Silva and Praum

4970 Bachelor's Thesis (3) Prereq.: Senior standing in computer science

Only students who have previously worked with a faculty member in a research group may register for Bachelor's Thesis credit, and then only with the permission of the faculty member. An undergraduate thesis is a publication-quality description of work done in previous semesters. At a minimum a thesis must be published as a technical report; ideally, it should be submitted to a conference or journal. A Bachelor's Thesis is intended as an alternative to the senior Software engineering Laboratory for students who are headed for graduate school.

4991 CE Senior Thesis I (2, F) Cross-listed as ECE 4991. Prereq.: CPSC/ECE 3992 and approved senior thesis proposal

Students work on original senior thesis project under the direction of their approved thesis advisor. This course along with CPSC/ECE 4992 substitute for CPSC/ECE 4710 (Computer Engineering Senior Project) for students who have chosen to do a thesis.

4992 CE Senior Thesis II (2, S) Cross-listed as ECE 4992. Prereq.: CPSC/ECE 4991

Students work on an original senior thesis project under the direction of their approved thesis advisor, make an oral presentation at the annual student technical conference, and prepare and submit their senior thesis for approval. This course along with CPSC/ECE 4991 substitute for CPSC/ECE 4710 (Computer Engineering Senior Project) for students who have chosen to do a thesis.

4999 Honors Thesis/Project (3) Upper-division Communications/Writing

Restricted to students in the Honors Program working on their Honors degree.

5010 Software Practice (4, FS) Prereq.: CPSC 2020 and permission of instructor

Meets with CPSC 3500. This course is for graduate students from other than the School of Computing. Practical exposure to the process of creating large software systems, including requirements specifications, design, implementation, testing, and maintenance. Emphasis on software process, software tools (debuggers, profilers, source code repositories, test harnesses), software engineering techniques (time management, code and documentation standards, source code management, object-oriented analysis and design), and team development practice. Much of the work will be in groups and will involve modifying preexisting software systems.

5020 Advanced Algorithms and Data Structures (3, FS) Prereq.: CPSC 5010 and permission of instructor

Meets with CPSC 3510. This course is for graduate students from other than the School of Computing. Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CPSC 2020. Balanced trees, heaps, hash tables, string matching, graph algorithms, external sorting and searching. Dynamic programming, exhaustive search. Space and time complexity, derivation and solution of recurrence relations, complexity hierarchies, reducibility, NP completeness.

5100 Foundations of Computer Science (3, S) Prereq.: CPSC 3100, CPSC 3510

Meets with CPSC 6100. Finite Automata and related topics (BDDs, Presburger Arithmetic, and decidable fragments of first-order logic). Automata on Infinite Words, connections with Specification and Verification of Systems. Push Down Automata, Turing Machines, Proofs by Reduction, Diagonalization, Problems in Computability. First-order Logic and Decidability. NP Completeness, P-space Completeness.

5210 Advanced Scientific Computing I (3, F) Prereq.: CPSC 3200, CPSC 3510, MATH 3160

Meets with CPSC 6210. An introduction to existing classical and modern numerical methods and their algorithmic development and efficient implementation. Topics include: numerical linear algebra, interpolation, approximation methods and parallel computation methods for nonlinear equations, ordinary differential equations, and partial differential equations.

5300 Artificial Intelligence (3, S) Prereq.: CPSC 3510

Meets with CPSC 6300. Introduction to field of artificial intelligence, including heuristic programming, problem-solving, search, theorem proving, question answering, machine learning, pattern recognition, game playing, robotics, computer vision.

5310 Robotics (3, F) Cross-listed as ME EN 5220. Prereq.: CPSC 1000, MATH 2250, PHYCS 2210

Meets with CPSC 6310. The mechanics of robots, comprising kinematics, dynamics, and trajectories. Planar, spherical, and spatial transformations and displacements. Representing orientation: Euler angles, angle-axis, and quaternions. Velocity and acceleration: the Jacobian and screw theory. Inverse kinematics: solvability and singularities. Trajectory planning: joint interpolation and Cartesian trajectories. Statics of serial chain mechanisms. Inertial parameters, Newton-Euler equations, D'Alembert's principle. Recursive forward and inverse dynamics.

5320 Computer Vision (3, S) Prereq.: CPSC 3510, MATH 2210, MATH 2270

Meets with CPSC 6320. Basic pattern-recognition and image-analysis techniques, low-level representation, intrinsic images, "shape from" methods, segmentation, texture and motion analysis, and representation of 2-D and 3-D shape.

5340 Natural Language Processing (3, F) Prereq.: CPSC 3510

Meets with CPSC 6340. Computational models and methods for understanding written text. Introduction to syntactic analysis, semantic analysis, discourse analysis, knowledge structures, and memory organization. A variety of approaches are covered, including conceptual dependency theory, connectionist methods, and statistical techniques. Applications include story understanding, fact extraction, and information retrieval.

5350 Machine Learning (3, F) Prereq.: CPSC 3510; CPSC 5300/6300 recommended

Meets with CPSC 6350. Techniques for developing computer systems that can acquire new knowledge automatically or adapt their behavior over time. Topics include concept learning, decision trees, evaluation functions, clustering methods, explanation-based learning, language learning, cognitive learning architectures, connectionist methods, reinforcement learning, genetic algorithms, hybrid methods, and discovery.

5450 Computer Security (3, S) Prereq.: CPSC 3400

(This course will be offered as CPSC 5961 in Spring 2005.)

5460 Operating Systems (4, F) Prereq.: CPSC 3510, CPSC 3400

Characteristics, objectives, and issues concerning computer operating systems. Hardware/software interactions, process management, memory management, protection, synchronization, resource allocation, file systems, security, and distributed systems. Extensive systems programming.

5470 Compiler Principles and Techniques (4, S) Prereq.: CPSC 3100, CPSC 3510, CPSC 3400

Lexical analysis, top-down and bottom-up parsing, symbol tables, internal forms and intermediate languages, run-time environments, code generation, code optimization, semantic specifications, error detection and recovery. Use of software tools for lexical analysis and parsing.

5480 Computer Networks (3, F) Prereq.: CPSC 3510, CPSC 3400

Meets with CPSC 6480. A comprehensive study of the principles and practices of data communication and networks. Topics include: transmission media, data encoding, local and wide area networking architectures, internetwork and transport protocols (e.g., IPv4, IPv6, TCP, UDP, RPC, SMTP), networking infrastructure (e.g., routers, name servers, gateways), network management, distributed applications, network security, and electronic commerce. Principles are put into practice via a number of programming projects.

5520 Anatomy of a Modern Programming Language (3, S) Prereq.: CPSC 3520

Requirements, challenges, and techniques for designing a modern programming language, currently focusing on Java as a case study. Syntactic and lexical issues, semantic specification, modularity concepts, support for object-oriented programming, types and subtypes, type safety and security, portability, compilability, dynamic linking and loading, program evolvability, use of meta data (reflection), multi-threading, native code generation and linkage, generic types, persistence.

5530 Database Systems (3, F) Prereq.: CPSC 3500

Meets with CPSC 6530. Representing information about real world enterprises using important data models including the entity-relationship, relational and object-oriented approaches. Database design criteria, including normalization and integrity constraints. Implementation techniques using commercial database management system software. Selected advanced Topics such as distributed, temporal, active, and multi-media databases.

5540 Human/Computer Interaction (3, F) Prereq.: CPSC 3500

Meets with CPSC 6540. Fundamentals of input/output devices, user interfaces, and human factors in the context of designing interactive applications.

5600 Introduction to Computer Graphics (3, S) Prereq.: CPSC 3500, MATH 2250; Coreq.: CPSC 3510 recommended

Basic display techniques, display devices, and graphics systems. Homogeneous coordinates, transformations, and clipping. Introduction to lighting models. Introduction to raster graphics and hidden-surface removal.

5605 Honors Introduction to Computer Graphics (3, S) Prereq.: CPSC 3500 and Honors status

Basic display techniques, display devices, and graphics systems. Homogeneous coordinates, transformations, and clipping. Introduction to lighting models. Introduction to raster graphics and hidden-surface removal.

5610 Advanced Computer Graphics I (3, F) Prereq.: CPSC 5600

Meets with CPSC 6610. Interactive 3D computer graphics, polygonal representations of 3-D objects. Interactive lighting models. Introduction to interactive texture mapping, shadow generation, image-based techniques such as stencils, hidden-line removal, and silhouette edges. Introduction to image-based rendering, global illumination, and volume rendering.

5630 Scientific Visualization (3, F) Prereq.: CPSC 3510; CPSC 3200 or CPSC 5210 or MATH 5600

Meets with CPSC 6630. Introduction to the techniques and tools needed for the visual display of data. Students will explore many aspects of visualization, using a ”from concepts to results” format. The course begins with an overview of the important issues involved in visualization, continues through an overview of graphics tools relating to visualization, and ends with instruction in the utilization and customization of a variety of scientific visualization software packages.

5720 Analog Integrated Circuit Design (3, S) Cross-listed as ECE 5720. Prereq.: ECE 3110

Meets with CPSC 6720. Design of analog and mixed-signal CMOS integrated circuits. Fundamental building blocks for analog circuits, including the basic principles of op amp, current mirror, and comparator design. The basics of sample-and-hold circuits. Students complete integrated circuit design, simulation, layout, and verification using computer-aided design tools.

5740 Computer-Aided Design of Digital Circuits (3, S) Cross-listed as ECE 5740. Prereq.: CPSC/ECE 3700, CPSC 3510

Meets with CPSC 6740. Introduction to theory and algorithms used for computer-aided synthesis of digital integrated circuits. Topics include algorithms and representations for Boolean optimization, hardware modeling, combination logic optimization, sequential logic optimization and technology mapping.

5750 Synthesis and Verification of Asynchronous VLSI Systems (3) Cross-listed as ECE 5750. Prereq.: CPSC/ECE 3700, CPSC 3510

Meets with CPSC 6750. Introduction to systematic methods for the design of asynchronous VLSI systems from high-level specifications to efficient, reliable circuit implantations. Topics include specification, protocols, graphical representations, synthesis, optimization using timing information, and verification.

5830 VLSI Architecture (3, S) Cross-listed as ECE 5830. Prereq.: CPSC/ECE 3700, CPSC/ECE 3810

Meets with CPSC 6830. Project-based study of a variety of Topics related to VLSI systems. Use of field programmable gate arrays to design, implement, and test a VLSI project.

5940 Seminar (1-3)

Current Topics in computer science. May be repeated for credit.

5950 Independent Study (1–4)

5960–5969 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2004–05 academic year. Contact the faculty member in charge for details. Check the on-line schedule for a more current listing of offerings.

- **CPSC 5960 Introduction to Computer Networks** (3,F). Prof. Hollaar.
- **CPSC 5961 Graduate TA Workshop** (1,F). Prof. Carter.
- **CPSC 5961 Networked Game Design** (3,S). Prof. Carter.
- **CPSC 5963 Advanced Manufacturing** (3,F). Prof. Drake.

6010 Writing Research Proposals (2, S) Prereq.: Graduate standing in Computer Science

Fundamental aspects of writing computer science research proposals, including thesis, dissertation, and grant proposals. Form, style, substance, and marketing of effective proposals will be considered. Emphasis is placed on developing and presenting clear and compelling ideas. Substantial writing and class presentations is required of all participants. (This is a half-semester course.)

6020 Conducting, Publishing, and Presenting Early-Career Research (3) Prereq.: Graduate standing in Computer Science

This is an independent study offering designed to encourage beginning graduate students to conduct, publish, and present original research early in their graduate careers. A graduate student can earn credit for CPSC 6020 by having a first-authored paper accepted for publication in a top-tier journal or conference and by subsequently presenting the published work in a one-hour research colloquium. The research must be conducted while a graduate student at Utah; the paper must be accepted within two years of enrolling in the graduate program; the journal or conference must be approved by the student's graduate committee; the colloquium must be presented as soon as possible after the acceptance of the paper; and the student must complete these requirements and register for CPSC 6020 within three years of enrolling in the graduate program. CPSC 6020 may not be repeated for credit.

6100 Foundations of Computer Science (3, S) Prereq.: CPSC 3100, CPSC 3510

Meets with CPSC 5100. Graduate and honors students only. Extra work required.

6110 Formal Methods for System Design (3, S) Prereq.: CPSC 5100/6100

Study of methods for formally specifying and verifying computing systems. Specific techniques include explicit state enumeration, implicit state enumeration, automated decision procedures for first-order logic, and automated theorem proving. Examples selected from the areas of superscalar CPU design, parallel processor memory models, and synchronization and coordination protocols.

6210 Advanced Scientific Computing I (3, F) Prereq.: CPSC 3200, CPSC 3510, MATH 3160

Meets with CPSC 5210. Graduate and honors students only. Extra work required.

6220 Advanced Scientific Computing II (3, S) Prereq.: CPSC 5210/6210 or MATH 5600

A study of the numerical solution of two and three dimensional partial differential equations that arise in science and engineering problems. Topics include: finite difference methods, finite element methods, boundary element methods, multigrid methods, mesh generation, storage optimization methods, and adaptive methods.

6300 Artificial Intelligence (3, S) Prereq.: CPSC 3510

Meets with CPSC 5300. Graduate and honors students only. Extra work required.

6310 Robotics (3, F) Cross-listed as ME EN 6220. Prereq.: CPSC 1000, MATH 2250, PHYCS 2220

Meets with CPSC 5310. Graduate and honors students only. Extra work required.

6320 Computer Vision (3, S) Prereq.: CPSC 3510, MATH 2210, MATH 2270

Meets with CPSC 5320. Graduate and honors students only. Extra work required.

6340 Natural Language Processing (3, F) Prereq.: CPSC 3510

Meets with CPSC 5340. Graduate and honors students only. Extra work required.

6350 Machine Learning (3, F) Prereq.: CPSC 3510; CPSC 5300/6300 recommended

Meets with CPSC 5350. Graduate and honors students only. Extra work required.

6360 Virtual Reality (3, S) Prereq.: CPSC 5310/6310

Human interfaces: visual, auditory, haptic, and locomotory displays; position tracking and mapping. Computer hardware and software for the generation of virtual environments. Networking and communications. Telerobotics: remote manipulators and vehicles, low-level control, supervisory control, and real-time architectures. Applications: manufacturing, medicine, hazardous environments, and training.

6470 Advanced Topics in Compilation (3, F) Prereq.: CPSC 5470

Compilation of modern languages. Optimization techniques, register allocation and instruction scheduling, garbage collection, exception handling. Linkers and late-stage compilation and optimization.

6480 Data Communications and Networks (3, F) Prereq.: CPSC 3510, CPSC/ECE 3810

Meets with CPSC 5480. Graduate and honors students only. Extra work required.

6530 Database Systems (3, F) Prereq.: CPSC 3510

Meets with CPSC 5530. Graduate and honors students only. Extra work required.

6540 Human/Computer Interaction (3, F) Prereq.: CPSC 3510

Meets with CPSC 5540. Graduate and honors students only. Extra work required.

6610 Advanced Computer Graphics I (3, S) Prereq.: CPSC 5600 or CPSC 3610

Meets with CPSC 5610. Graduate and honors students only. Extra work required.

6620 Advanced Computer Graphics II (3, S) Prereq.: CPSC 5610/6610

Introduction to ray-tracing. Intersection methods for 3-D objects, reflection and refraction. Introduction to surface and solid texturing. Introduction to continuous-tone pictures and the aliasing problem. Special effects such as soft shadows, depth-of-field, motion-blur, and indirect lighting.

6630 Scientific Visualization (3, F) Prereq.: CPSC 3510; CPSC 3200 or CPSC 5210/6210 or MATH 5600

Meets with CPSC 5630. Graduate and honors students only. Extra work required.

6670 Computer-Aided Geometric Design I (3, F) Prereq.: MATH 2210, MATH 2250, CPSC 3510; Coreq.: CPSC 5600/6600

Introduction to current concepts and issues in CAGD systems with emphasis on free- form surface design; mathematics of free-form curve and surface representations, including Coons patches, Bezier method, B-splines, triangular interpolants, and their geometric consequences; classical surface geometry; local and global design tradeoffs and explicit and parametric tradeoffs; subdivision and refinement as techniques in modeling; current production capabilities compared to advanced research. Laboratory experiments with current CAD systems.

6680 Computer-Aided Geometric Design II (3) Prereq.: CPSC 6670

Project based on material covered in CPSC 6670.

6710 Digital VLSI Design (4, F) Cross-listed as ECE 6710. Prereq.: CPSC/ECE 3700

Introduction to basic concepts of the design of CMOS integrated circuits. Static and dynamic properties of CMOS circuits, composite layout of CMOS circuits, and modeling of transistors. Commonly encountered CMOS circuits. Students complete design, composite layout, and simulation of an integrated circuit project using computer-aided design tools.

6720 Advanced Integrated Circuit Design II (3, S) Cross-listed as ECE 6720. Prereq.: ECE 3100

Meets with CPSC 5720. Graduate and honors students only. Extra work required.

6721 Analog Integrated Circuits Lab (1, S) Cross-listed as ECE 6721. Coreq.: CPSC/ECE 5720/6720

Optional lab that accompanies CP SC/ECE 5720/6720. Students will test and characterize transistors, circuits, and systems on modern CMOS chips.

6722 Analog Integrated Circuits Project Testing (1, F) Cross-listed as ECE 6722. Prereq.: CPSC/ECE 5720/6720

This course is designed for students who fabricated an integrated circuit in CP SC/ECE 5720/6720. Students will test their chips independently and report on the experimental results.

6740 Computer-Aided Design of Digital Circuits (3, S) Cross-listed as ECE 6740. Prereq.: CPSC/ECE 3700, CPSC 3510

Meets with CPSC 5740. Graduate and honors students only. Extra work required.

6750 Synthesis and Verification of Asynchronous VLSI Systems (3, F) Cross-listed as ECE 6750. Prereq.: CPSC/ECE 3700, CPSC 3510

Meets with CPSC 5750. Graduate and honors students only. Extra work required.

6770 Advanced Digital VLSI Systems Design (3) Cross-listed as ECE 6770. Prereq.: CPSC/ECE 5710/6710 and instructor permission

Full custom, high speed, high performance CMOS circuit design issues, methodologies, and techniques. Failure modes, modeling techniques, testing, clock skew analysis, clock distribution, power analysis, power line distribution, electrical rules checking, megacell design flow, and other important design issues.

6810 Computer Architecture (3, F) Cross-listed as ECE 6810. Prereq.: CPSC/ECE 3700, CPSC/ECE 3810

Principles of modern high performance computer and micro architecture: static vs. dynamic issues, pipelining, control and data hazards, branch prediction and correlation, cache structure and policies, cost-performance and physical complexity analyses.

6830 VLSI Architecture (3, S) Cross-listed as ECE 6830. Prereq.: CPSC/ECE 3700, CPSC/ECE 3810

Meets with CPSC 5830. Graduate and honors students only. Extra work required.

6930–6944 Seminar (1-3)

Current Topics in Computer Science. May be repeated for credit.

6950 Independent Study (1–4)

6960–6969 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2004–2005 academic year. Contact the instructor for details. Check the on-line schedule for a more current listing of offerings.

- **CPSC 6960 Digital Intellectual Property Law** (2,F). Prof. Hollaar.
- **CPSC 6961 Creating Complex Software Systems** (3,F). Nathan Dykman.
- **CPSC 6962 Advanced Algorithms** (3,F). Prof. Silva.
- **CPSC 6963 VLSI Logic Test Validation and Verification** (3,F). Prof. Kalla.
- **CPSC 6963 Multiagent Systems** (3,S). Prof. Henderson.
- **CPSC 6964 High Performance Computing** (3,S). Prof. Berzins.
- **CPSC 6965 Visual Computing** (3,S). Prof. Cohen.
- **CPSC 6966 Network Security** (3,S). Prof. Kasera.

6970 Masters Thesis Research (1–12)

6980 Faculty Consultation Masters (1–12)

7120 Information-Based Complexity (3) Prereq.: CPSC 3200, MATH 2270, MATH 3210

Analysis of optimal computational methods for continuous problems. Introduction to the general worst case theory of optimal algorithms, linear problems, and spline algorithms as well as selected nonlinear problems. Examples include optimal integration, approximation, nonlinear zero finding, and fixed points.

7240 Sinc Methods (3, S) Prereq.: CPSC 5210/6210 or MATH 5600 or MATH 5610

Sinc methods for solving difficult computational problems, such as partial differential and integral equation problems, that arise in science and engineering research. Emphasis on parallel computation. Applications vary, depending on participants in the class. Students are given projects—whenever possible in their areas of research—that lead to publishable research articles.

7310 Advanced Robotics (3, S) Cross-listed as ME EN 7230. Prereq.: CPSC/ME EN 5310/6310 5220/6220

Covers the kinematics, dynamics, and control of robotic manipulators. Projects controlling robots will be an integral part of the course.

7420 Embedded Systems (3, S)

(Alternate years. Will be offered Spring 2005 as CPSC 7963.)

7430 Concurrency (3, S)

(Alternate years. Next planned offering is Spring 2006.)

7460 Distributed Operating Systems (3) Prereq.: CPSC 5460, CPSC 5480/6480

Practical distributed operating systems concepts from basics through the state of the art. Topics include interprocess communication, client-server systems, distributed shared memory, distributed file systems, distributed databases, portable computing, software fault tolerance, and wide-area (e.g. web) applications. Work includes individual oral presentations, a group project, and a written research report.

7520 Programming Language Semantics (3, S) Prereq.: CPSC 3520, CPSC 3100

Examination of the formal and pragmatic ideas behind programming language design. Imperative, functional, logic, object-oriented, and multi-paradigm languages. Lambda calculus, fixpoints, type systems, and predicate logic. Denotational semantics and models of concurrency. (Alternate years. Next planned offering is Spring 2005.)

7530 Functional Programming (3, S) Prereq.: CPSC 3520, CPSC 3100

(Alternate years. Next planned offering is Spring 2006.)

7650 Image Synthesis (3, F) Prereq.: CPSC 5620/6620, CPSC 6670, MATH 5010

Using camera and sensor simulation along with physical simulation to generate realistic synthetic images.

7810 Advanced Computer Architecture (3, S) Cross-listed as ECE 7810. Prereq.: CPSC/ECE 5810/6810

Issues in the design of modern microprocessors, with emphasis on current research topics in the field. Offered in alternate years.

7820 Parallel Computer Architecture (3, S) Cross-listed as ECE 7820. Prereq.: CPSC/ECE 5810/6810

Architecture, design, and analysis of parallel computer systems: vector processing, data vs. control concurrency, shared memory, message passing, communication fabrics, case studies of current high performance parallel systems. Offered in alternate years.

7930–7944 Advanced Seminar (1–3)

May be repeated for credit.

7950 Independent Study (1–4)

7960–7969 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2004–05 academic year. Contact the faculty member in charge for details. Check the on-line schedule for a more current listing of offerings.

- **CPSC 7961 Advanced Topics on Flow Visualization** (3,F). Dr. Tricoche.
- **CPSC 7962 Embedded Systems** (3,S). Prof. Regehr.
- **CPSC 7963 Functional Programming** (3,S). Prof. Flatt.
- **CPSC 7964 Vision Science** (3,S). Prof. Thompson.
- **CPSC 7965 Advanced Database Systems** (3,S). Prof. Freire.
- **CPSC 7966 Vision Science** (3,S). Prof. Whitaker.
- **CPSC 7967 Advanced Topics in Scientific Computing** (3,S). Prof. Sikorski.
- **CPSC 7968 Parallel Computer Organization** (3,S). Prof. Balasubramonian

7970 PhD Dissertation Research (1–12)

7980 Faculty Consultation PhD (1–12)

4 School of Computing Faculty and Their Research Interests

Rajeev Balasubramonian
Assistant Professor, School of Computing
Ph.D., University of Rochester, 2003

Professor Balasubramonian[1] joins the faculty in Fall 2003. His research interests include high-performance and low-power computer architecture. Broadly, he is interested in employing large transistor budgets in a complexity-effective manner to solve a diverse set of problems. The primary focus of his research has been the evaluation of technology trends and their impact on the design of next-generation microprocessors. He is especially concerned with problems involving long on-chip wire delays and memory latencies, which represent two of the most important bottlenecks in the future. Among other things, his dissertation studied scalability issues in clustered processors, which are designs that incorporate many processing units on a single die at low complexity. His future research goals include evaluations of multi-threaded workloads, heterogeneity within a clustered system, on-chip interconnect networks, power, and thermal issues.

Martin Berzins
Professor, School of Computing
Ph.D., University of Leeds, 1981

Professor Berzins[2] joined the faculty in 2003. His research area is the study of novel computational algorithms for the numerical solution of partial differential equations on both serial and parallal computers. In particular, his interests are in devising new adaptive mesh algorithms, exploiting new parallel computing techniques to reduce computing times, and to investigate novel applications in the areas of environmental and other complex systems.

[1] http://www.cs.utah.edu/~rajeev/
[2] http://www.cs.utah.edu/~mb/

Erik Brunvand
Associate Professor, School of Computing
Ph.D., Carnegie Mellon University, 1991

Professor Brunvand[3] joined the faculty in 1990. He has interests in computer architecture and VLSI systems in general, and self-timed and asynchronous systems in particular. One aspect of his research involves compiling concurrent communicating programs into asynchronous VLSI circuits. The current system allows programs written in a subset of Occam, a concurrent message-passing programming language based on CSP, to be automatically compiled into a set of self-timed circuit modules suitable for manufacture as an integrated circuit. He is also interested in investigating the effects of asynchrony on computer systems architecture at a higher level. To explore these ideas he is building a series of prototype asynchronous computer systems out of FPGA and custom VLSI chips.

John Carter
Associate Professor, School of Computing
Ph.D., Rice University, 1992

Professor Carter[4] joined the faculty in January 1993. His research interests include computer architecture, operating systems, distributed systems, and computer networks. Of particular interest are novel memory system designs, both hardware and software. Dr. Carter is co-leading two research projects: the Impulse Adaptable Memory Systems project and the Khazana project. The goal of the Impulse project is to attack the primary problem limiting performance in future computer systems – the inability of conventional memory systems to supply data fast enough to avoid processing stalls – by developing a main memory controller and associated software that allows applications to dynamically change the way that the processor's memory hierarchy is managed. Khazana makes it easier for programmers to develop sophisticated distributed applications by addressing the shared state management problem faces by most such applications. Khazana exports the abstraction of a distributed secure persistent globally shared store that applications can use to store their shared state. It is responsible for performing many of the common operations needed by distributed applications, including replication, consistency management, and fault recovery.

Elaine Cohen
Professor, School of Computing
Adjunct Associate Professor of Mathematics
Ph.D., Syracuse University, 1974

Professor Cohen[5] has held a faculty position since 1974. Currently she is co-head of the School's Computer-Aided Geometric Design Group. Present research is centered around representational and algorithmic problems associated with geometric modeling, graphics, scientific visualization physically based modeling, process planning, CAD/CAM and CAE. Dr. Cohen received a B.A. in Mathematics from Vassar College in 1968 and M.S. and Ph.D. degrees in mathematics from Syracuse University in 1970 and 1974, respectively.

[3] http://www.cs.utah.edu/~elb/
[4] http://www.cs.utah.edu/~retrac/
[5] http://www.cs.utah.edu/~cohen/

Al Davis
Professor, School of Computing
Ph.D., University of Utah, 1972

Professor Davis[6] joined the faculty in 1993. His research interests involve high performance computer architectures and digital system design methodologies. More specifically he is interested in parallel processor architectures, high performance uniprocessor I/O architecture, VLSI, VLSI CAD, high performance communication, and asynchronous circuits. Prior to his joining the faculty in the fall of 1993, he spent the previous 12 years as a research scientist working on the design and implementation of parallel processing systems at Schlumberger Palo Alto Research and subsequently at Hewlett-Packard Laboratories. Recent accomplishments include 1) the development of an automatic asynchronous circuit synthesis system called STETSON; 2) the design and implementation of an asynchronous scalable parallel communication fabric VLSI component called FEDEX which is capable of supporting 500 MB/sec sustained bandwidth on each of its 7 ports; and 3) the development of an extensible and scalable parallel processing system called MAYFLY and 4) the development of a very low latency message passing protocols called Direct Deposit. He is currently involved in the DARPA sponsored Impulse project (Prof. John Carter is the PI) which is developing an adaptive memory controller that is capable of dynamically organizing cache lines to suit the applications needs. During the 1999-2000 academic year, Professor Davis was on sabbatical at Intel in Austin, Texas where he lead the I/O and memory architecture efforts for the IA32 processor which is expected to be in production in 2003.

Matthew Flatt
Assistant Professor, School of Computing
Ph.D., Rice University, 1999

Professor Flatt's[7] research interests cover all practical and theoretical aspects of programming languages, systems, and environments. As part of the Programming Languages Team (PLT), he is one of the principal architects of the DrScheme programming environment and a co-author of the *How to Design Programs* textbook. His current research topics include module languages for software components, object-oriented languages for classes and mixins, and high-level operating systems for cooperating applications.

Ganesh C. Gopalakrishnan
Professor, School of Computing
Ph.D., State University of New York at Stony Brook, 1986

Professor Gopalakrishnan's[8] primary research is in verification methods for concurrent systems such as shared memory systems, microprocessor busses, multithreaded software, and message passing networks. He also maintains active interest in self-timed design. Today's concurrent systems employ complex protocols that are expected to guarantee properties such as in-order arrival of messages, deadlock freedom, and liveness. In modern design approaches, these systems are subject to a battery of conventional tests, and when all these pass, model-checking methods are brought to bear to tracking down elusive bugs that may cripple the system well after field deployment. The effectiveness of conventional model-checking methods is limited by their inability to handle large state spaces, deal with parameterized designs, or provide guidelines for writing a comprehensive list of properties to check. Our group's recent efforts have addressed these problems using realistic driving problems such as generalized multi-level PCI I/O busses, the Intel Itanium Shared Memory Model, and the Java Shared Memory Model for Multithreading. Professor Gopalakrishnan was a general co-Chair of the Internal Symposium on Formal Methods in Computer-Aided Design (FMCAD) in November 1998, the International Symposium on Advanced Research in Asynchronous Circuits and Systems (Async) in November 1994. He organized the Workshop on Advances in Verification (WAVe) as well as the workshop on Formal Specification and Verification Methods for Shared Memory Systems, both in year 2000.

[6] http://www.cs.utah.edu/~ald/
[7] http://www.cs.utah.edu/~mflatt/
[8] http://www.cs.utah.edu/~ganesh/

David H. Hanscom
Clinical Professor, School of Computing
Ph.D., Case Western Reserve University, 1970

Professor Hanscom's[9] background is in the field of communications processor design at Sperry Univac, where he worked from 1970 to 1982. Since then he has been responsible for administering the undergraduate Computer Science and Computer Engineering programs at the University of Utah. In that capacity he teaches core computer science classes, serves as faculty advisor for individual students and for the Student Chapter of the Association for Computing Machinery, participates in student recruiting activities, and serves as director of the High School Computing Institute. His interests are in the areas of undergraduate education, computer architecture, and data communications.

Charles Hansen
Associate Professor, School of Computing
Ph.D., University of Utah, 1987

Professor Hansen[10] joined the faculty in 1998. His research interests span scientific visualization, computer graphics, and high performance computing. Scientific visualization of large scale problems is of key interest and recent work involves taking into consideration time-varying data and exploiting this for speeding up the visualization process. Other methods for visualizing large amounts of data are multiresolution models and view dependent algorithms. The interest in computer graphics is driven from the scientific visualization perspective but includes parallel algorithms for speeding up global illumination.

Thomas C. Henderson
Professor, School of Computing
Ph.D., University of Texas, 1979

Professor Henderson's[11] professional interests include autonomous agents, multisensor systems, and simulation. Major areas of current research are robot behavior specification, simulation, multisensor integration, and bio-based computational models. Prior to his arrival at Utah, he was a visiting professor at the Institut National de Recherche en Informatique et en Automatique (INRIA), France, and a Research Associate at the Institut fuer Nachrichtentechnik, Deutsche Forschungs und Versuchsanstalt fuer Luft und Raumfahrt (DFVLR), Germany.

Lee A. Hollaar
Professor, School of Computing
Ph.D., University of Illinois at Urbana-Champaign, 1975

Professor Hollaar's[12] primary interest is in legal issues regarding computers, particularly the intellectual property protection of software and information. As a Fellow with the Committee on the Judiciary, he has advised the United States Senate on computer-related issues such as encryption, copyright and patent, and regulation of the internet. He was one of the drafters of the Utah Digital Signature Act, the first law in the world to legally recognize digital signatures, and is active in the implementation of the required infrastructure. He directed the Utah Retrieval System Architecture (URSA) project, which developed hardware and software systems to support large information retrieval systems, including a special-purpose VLSI processor for the rapid searching of text and one of the first workstation-based client-server distributed systems for information retrieval. He was also the University's director of campus networking, and continues to work in communications networks and distributed systems.

[9] http://www.cs.utah.edu/~hanscom/
[10] http://www.cs.utah.edu/~hansen/
[11] http://www.cs.utah.edu/~tch/
[12] http://www.cs.utah.edu/~hollaar/

John M. Hollerbach
Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1978

Professor Hollerbach's[13] interests include robotics and virtual reality. The focus in virtual reality is on improving the transparency and sense of immersion through better mechanical interfaces and their control, better visual and auditory displays, and sensorimotor integration. Haptic interfaces are being employed for virtual manipulation of mechanical CAD models and for scientific visualization. The Treadport locomotion interface is being employed for walk-through synthetic environments such as outdoor terrain.

Christopher R. Johnson
Professor, School of Computing
Research Associate Professor of Physics and Bioengineering
Ph.D., University of Utah, 1989

Professor Johnson's[14] research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods for partial differential equations, numerical analysis, problem solving environments, computational problems in medicine, and scientific visualization. In 1992, Professor Johnson was awarded a Young Investigator's Award from the NIH, in 1994 he was awarded the National Young Investigator (NYI) Award from the NSF, and in 1995 he was awarded the Presidential Faculty Fellow (PFF) Award from the NSF. In 1996 he received a DOE Computational Science Award and in 1997 received the Par Excellence Award from the University of Utah Alumni Association and the Presidential Teaching Scholar Award. In 1999, Professor Johnson was Awarded the Governor's Medal for Science and Technology from Utah Governor Michael Leavitt. He directs the Scientific Computing and Imaging Institute.

Sneha Kasera
Assistant Professor, School of Computing
Ph.D., University of Massachusettes Amherst, 1999

Professor Kasera[15] joined the faculty in July 2003. Prior to coming to the University of Utah, he spent four exciting years in the Mobile Networking research department of Bell Labs in New Jersey. His research interests include computer networks and systems encompassing mobile and pervasive systems and wireless networks, network security and reliability, overload and congestion control, multicast communication, Internet pricing, Internet measurements and inferencing. He heads the Advanced Networked Systems Research (ANSR) group that he recently started.

Robert Kessler
Professor, School of Computing
Ph.D., University of Utah, 1981

Professor Kessler's[16] current research interests are in agents, software engineering, distributed systems, and visual programming. In the early 90's, he founded the Center for Software Science, a state of Utah Center of Excellence. He has also founded several startup companies and is currently involved with an Internet startup company, emWare, as a member of the board. He has served as member-at-large of ACM SIGPLAN and Vice Chairman for Conferences for SIGPLAN. He recently completed a seven year assignment as co-editor-in-chief of the International Journal of Lisp and Symbolic Computation.

[13] http://www.cs.utah.edu/~jmh/
[14] http://www.cs.utah.edu/~crj/
[15] http://www.cs.utah.edu/~kasera/
[16] http://www.cs.utah.edu/~kessler/

Mike Kirby
Assistant Professor, School of Computing
Ph.D., Brown University, 2002

Professor Kirby[17] joined the faculty in 2002. His research focus is on large-scale scientific computation and visualization, with an emphasis on the scientific cycle of mathematical modeling, computation, visualization, evaluation, and understanding. His primary research interests include computational science and engineering, high-order numerical methods, concurrent programming, scientific visualization, and high performance computing.

Arthur H. Lee
Clinical Associate Professor, School of Computing
Ph.D., University of Utah, 1992

Professor Lee[18] joined the faculty in 2001. His interests revolve around object-oriented programming. Of particular interest are software design, software architecture, and software evolution. These are being experimented in areas such as programming languages, database systems, and distributed systems. He is also interested in improving computer science curricula. He tries to bring emerging ideas from research and industry into the classroom, always looking for new ones to incorporate into the CS curricula. Some of the emerging ideas that Professor Lee is investigating include aspect-oriented programming, web programming, and embedded programming. He is also interested in finding innovative and effective ways to use computers and other technologies in teaching.

Jay Lepreau
Research Professor, School of Computing
B.S., University of Utah, 1983

Professor Lepreau's[19] research interests focus on operating systems, but expand into many other areas including security, networking, component software, programming and domain-specific languages, compilers, distributed systems, and software assurance. As head of the Flux Research Group, he currently leads three DARPA and NSF-sponsored research projects. The "Alchemy" project is developing a new model of component programming for embedded and other low-level systems. Utah's Active Networks effort is attempting to develop a router OS that can safely and speedily "execute" Java bytecode-carrying packets. Finally, in a related effort his group is constructing a unique research instrument: a remotely configurable 1000-node network testbed and emulation facility. In these efforts, his group has developed much software, including the "Janos" active network OS, the "Knit" component composition language, the OSKit, the Flick IDL compiler, the Fluke/Flask OS, and the "Alta" and "KaffeOS" Java operating systems. In 1994 he founded the prestigious *Usenix/ACM/IEEE Symposium on Operating Systems Design and Implementation* (OSDI) conference series, and served as its first program chair.

Gary E. Lindstrom
Professor, School of Computing
Ph.D., Carnegie-Mellon University, 1971

Professor Lindstrom's[20] research interests include programming languages, databases, and scientific data management. He is on the editorial board of *International Journal of Parallel Programming*, and was Editor-in-Chief from its founding until 1993. With Doug DeGroot, he co-edited the book *Logic Programming: Functions, Relations and Equations* published by Prentice-Hall. Professor Lindstrom has been a member of the National Science Foundation Computer and Computation Research Advisory Committee, and served as a Distinguished Visitor of the IEEE Computer Society. In 1981 he received the College of Engineering Outstanding Teaching Award.

[17] http://www.cs.utah.edu/~kirby/
[18] http://www.cs.utah.edu/~alee/
[19] http://www.cs.utah.edu/~lepreau/
[20] http://www.cs.utah.edu/~gary/

Steven Parker
Research Assistant Professor, School of Computing
Ph.D., University of Utah, 1999

Professor Parker's[21] research is in problem solving environments, which tie together scientific computing, scientific visualization, and computer graphics. As a member of the Scientific Computing and Imaging (SCI) Institute, he is the principal architect of the SCIRun Software System, which formed the core of his Ph.D. dissertation. He is currently the chief architect of Uintah, a software system designed to simulate accidental fires and explosions using thousands of processors, and the rtrt/manta interactive ray tracing system. He was a recipient of the Computational Science Graduate Fellowship from the Department of Energy.

Emil Praun
Assistant Professor, School of Computing
Ph.D., Princeton University, 2002

Professor Praun[22] joined the faculty in 2002. His main interests are in computer graphics. In particular, he has been working on applications of local parameterizations of triangle meshes: dividing 3D-embedded surfaces into collections of patches and ?unfolding? each of the patches into the plane. He is also interested in combining direct acquisition of 3D surfaces with procedural detail generation methods. He is a recipient of the John E. and Marva M. Warnock Presidential Endowed Chair for Faculty Innovation in Computer Science.

John Regehr
Assistant Professor, School of Computing
Ph.D., Virginia, 2001

Professor Regehr's[23] background is in operating systems and real-time systems, and his current research focuses on techniques and tools that make it easier and faster to develop efficient, reliable embedded software. He joined the faculty in 2003, having received his Ph.D. from the University of Virginia in 2001. Professor Regehr was previously a postdoc in the School of Computing from 2001-2003 and an adjunct assistant professor in the School of Computing during 2002-2003.

Richard F. Riesenfeld
Professor, School of Computing
Ph.D., Syracuse University, 1973

Professor Riesenfeld[24] has been involved in research in the areas of computer graphics, animation, computer aided geometric design and CAD/CAM since joining the faculty in 1972. Recently he has been investigating a broad spectrum of research problems in computer graphics, geometric modeling, and manufacturing within an integrated experimental testbed system motivated by the unifying principles of spline theory.

Ellen M. Riloff
Associate Professor, School of Computing
Ph.D., University of Massachusetts at Amherst, 1994

Professor Riloff[25] joined the faculty in 1994. Her research interests are in natural language processing, machine learning, and artificial intelligence. She is particularly interested in techniques for automatically generating dictionaries and knowledge bases for natural language processing. Most of her work revolves around the task of *information extraction*, which involves extracting information from text. The NLP research group at Utah has built its own NLP system called Sundance, which is a partial parser that activates and instantiates case frames for information extraction. Current research projects include bootstrapping techniques for learning extraction patterns, corpus-based techniques for learning semantic dictionaries, and corpus-based methods for coreference resolution.

[21] http://www.cs.utah.edu/~sparker/
[22] http://www.cs.utah.edu/~emilp/
[23] http://www.cs.utah.edu/~regehr/
[24] http://www.cs.utah.edu/~rfr/
[25] http://www.cs.utah.edu/~riloff/

Peter Shirley
Associate Professor, School of Computing
Ph.D., University of Illinois, 1991

Professor Shirley[26] joined the faculty in 1996. He is interested in creating highly realistic images of virtual environments, and visualization of complex data. The former involves explicit and procedural generation of geometric models with realistic optical characteristics, light transport simulation to determine the outgoing light distribution from surfaces, and tone reproduction to create images displayable on low dynamic range media such as paper and CRTs. The latter involves issues of visual representation of complex data, as well as strategies for navigation and interaction that help the user extract local and global information about the data. In 2000 Professor Shirley received the University of Utah Student's Choice Teaching Award and in 1998 he received the College of Engineering Outstanding Teacher Award.

Kris Sikorski
Professor, School of Computing
Ph.D., University of Utah, 1982

Professor Sikorski's[27] current research interests are in the areas of distributed parallel scientific computation and computational complexity with emphasis on information based complexity. Of specific interest are applied problems in geophysics (3-D modeling of earthquakes), combustion (fluid mechanics), and electromagnetic wave propagation (Maxwell equations). Various parallel explicit and implicit algorithms are being studied and implemented on massively parallel machines. Information based complexity is a study of optimal algorithms for problems which are approximately solved, because of partial and contaminated information. Optimal algorithms for solving nonlinear problems with use of various error criteria are of special interest to Professor Sikorski.

Claudio Silva
Associate Professor, School of Computing
Ph.D., SUNY Stonybrook, 1996

Professor Silva[28] joined the faculty in 2003. His main research interests are in graphics, visualization, applied computational geometry, and high-performance computing. His current research focuses on architectures and algorithms for building scalable displays, rendering techniques for large datasets, 3D scanning, and algorithms for graphics hardware, and includes recent work on 3D compression and multi-resolution techniques for point-sets and polygonal surfaces, large-scale 3D photography, visibility culling and volume rendering.

Konrad Slind
Assistant Professor, School of Computing
Ph.D., TU Munich, 1999

Professor Slind[29] joined the faculty in 2001. His research interests are in logic and functional programming. He is particularly interested in higher order logic, its implementation, and its application to deductive verification of system properties. Recent research has investigated the modeling of generic and functional programming. Before coming to Utah, he participated in a European project that developed middleware for applications that require a theorem proving component.

[26] http://www.cs.utah.edu/~shirley/
[27] http://www.cs.utah.edu/~sikorski/
[28] http://www.cs.utah.edu/~csilva/
[29] http://www.cs.utah.edu/~slind/

Frank Stenger
Professor, School of Computing
Ph.D., University of Alberta, 1965

Professor Stenger's[30] research interests include the development of new methods of computation and the computer solution of computationally difficult problems from science and engineering, such as inverse problems, crack problems, flow problems and heat problems. He developed the Sinc methods, which provide optimal algorithms in all areas of engineering computation. He is currently writing, jointly with Michael O'Reilly and Tao Zhang, a *Sinc Tool Box* computer-based tutorial package to make these methods accessible to users. One of his students (Kenneth Parker) has recently completed his Ph.D. dissertation *PTOLEMY: A Sinc-Collocation Mapping Sub-System*, which is a computer sub-package of *Maple* that automates the solution of partial differential equations. Several of his students have recently written or are presently writing computer packages for solving a broad range of difficult engineering problem—such as Navier-Stokes equations (Barkey and Vakili, Narasimhan), Maxwell equations (Naghsh-Nilchi) based on his recently discovered Sinc method of approximating indefinite convolutions, which leads to a unified approach for solving elliptic, parabolic, and hyperbolic differential equations. Another student (Ross Schmittlein) is writing a package based on Sinc, constructing conformal maps. Stenger and his former student O'Reilly have been developing methods which make it possible to invert the Helmholtz equation without computing the forward solution. During the next few academic years he expects to extend these inversion methods so that they can be applied to rendering, to visualization, to the determination of paths for robots, and to the inversion of heat and electromagnetic problems for medical and geophysical applications. Seven of his papers have been accepted for publications during this past academic year.

Cynthia A. Thompson
Assistant Professor, School of Computing
Ph.D., University of Texas, 1998

Professor Thompson[31] joined the faculty in 2000. Her research interests are in machine learning, natural language processing, and artificial intelligence. She is especially interested in applying machine learning to complex, relational tasks such as language processing and adaptive interfaces. After receiving her Ph.D., she spent two years at the Center for the Study of Language and Information at Stanford University, building and studying conversational interfaces that adapt themselves over the course of several interactions with a user.

William B. Thompson
Professor, School of Computing
Ph.D., University of Southern California, 1975

Professor Thompson's[32] primary research interest is in the area of visual perception. Currently, he is exploring how an understanding of computational vision can aid in improving the spatial information conveyed by graphical displays. He is also active in the exploration of techniques for analyzing visual motion, sensing for manufacturing, and vision-based navigation. Professor Thompson joined the faculty in 1991 after 16 years in the Computer Science Department at the University of Minnesota.

Ross Whitaker
Associate Professor, School of Computing
Ph.D., University of North Carolina, 1993

Professor Whitaker[33] joined the faculty in 2000. He has interests in computer vision, visualization, and image processing. In the area of medical image processing, Dr. Whitaker is a codeveloper of the "Insight" toolkit for segmenting and visualization the 3D color data associated with the Visible Human Project. Dr. Whitaker is also working on new, statistics-based methods for building surface models from noisy range measurements, such as those from laser radar and ultrasound. In the area of visualization, Dr. Whitaker is developing new methods for visualizing biological volume datasets and for processing the surface models that are derived from these datasets.

[30] http://www.cs.utah.edu/~stenger/
[31] http://www.cs.utah.edu/~cindi/
[32] http://www.cs.utah.edu/~thompson/
[33] http://www.cs.utah.edu/~whitaker/

Joseph L. Zachary
Clinical Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1987

Professor Zachary[34] joined the faculty in 1987. He is interested in finding ways to use computers in teaching science and engineering in general, and computer science in particular. He is currently developing Web-based tools and curricula for teaching introductory programming, computer science, and scientific computing courses. The author of two textbooks in scientific programming, Professor Zachary received the 1999 IEEE Computer Science and Engineering Undergraduate Teaching Award, won the University of Utah Distinguished Teaching Award in 1997, was named a Department of Energy Undergraduate Computational Science Education Award winner in 1996, was a University of Utah Presidential Teaching Scholar in 1995, and received the College of Engineering Outstanding Teaching Award in 1990.

[34] http://www.cs.utah.edu/~zachary/