

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

2001–2002 Academic Year

June 29, 2001

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 math while exposing them to the breadth of issues that arise in computer science. Students then take five core courses in software engineering, computer architecture, algorithms and data structures, software systems, and theory. Students 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, software systems, 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 undergraduate curriculum encompasses a general education in mathematics, science, and the humanities.

The School also offers 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 Engineering. Information about that program is available in a separate handbook or from the web page referenced below.

The University of Utah is committed to policies of equal opportunity, affirmative action, and nondiscrimination. The University seeks to provide equal access to its programs, services, and activities for people with disabilities. Reasonable prior notice is needed to arrange accommodations.

(This handbook is available online at <http://www.cs.utah.edu/dept/handbooks>.)

Faculty

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

Administration

| | | |
|------------------------|---|----------|
| Thomas C. Henderson | Director | 581-3601 |
| Charles Hansen | Associate Director | 581-3154 |
| Dave Hanscom | Director, Undergraduate Studies | 581-7023 |
| Al Davis | Director, Graduate Studies | 581-3991 |
| Christopher R. Johnson | Director, Computational Engineering and Science | 581-7705 |
| Chris Myers | Director, Computer Engineering | 581-6490 |
| Joseph L. Zachary | Director, Educational Programs | 581-7079 |

Administrative Staff

| | | |
|---------------|-------------------------------|----------|
| Shawn Darby | School Administrative Manager | 581-3950 |
| Erin Davies | Administrative Assistant | 581-8226 |
| Monica Heaton | Administrative Officer | 581-9371 |
| Ann Torrence | Program Development Director | 581-7631 |

Academic Advising

| | | |
|--------------|----------------------------------|----------|
| Holly Gough | Graduate Coordinator | 585-3551 |
| Sandy Hiskey | Undergraduate Academic Counselor | 581-8224 |

Technical Staff

| | | |
|--------------|------------------------------------|----------|
| Jim Conforti | NT Lab Manager | 585-9047 |
| Corey Hatch | Digital Systems Laboratory Manager | 581-7845 |
| Chad Lake | Computing Facility Director | 585-7614 |

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

Contents

| | |
|---|------------|
| Faculty | iii |
| Administration and Staff | v |
| Contents | vii |
| 1 The Computer Science Major and Minor | 1 |
| 1.1 Becoming a Computer Science Major | 1 |
| 1.2 Undergraduate Advising | 2 |
| 1.3 Requirements for the Bachelor of Science Degree | 2 |
| 1.4 Computer Science Honors Degrees | 5 |
| 1.5 Computer Science as a Minor | 6 |
| 1.6 Students Pursuing a Second Degree or a Double Major | 6 |
| 1.7 Course Enrollment Priorities for Non-Majors | 6 |
| 1.8 Undergraduate Financial Assistance | 7 |
| 1.9 Employment Opportunities | 7 |
| 1.10 Student Participation in School Affairs | 8 |
| 2 Computing Facilities | 9 |
| 3 Computer Science Courses | 11 |
| 4 School of Computing Faculty and Their Research Interests | 23 |

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 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.

Only computer science majors, computer engineering majors, and computer science minors can take upper-division (3000 or higher) computer science classes. Advancement to full major status is a competitive process. To be considered for full major status in computer science, a student must have:

1. A minimum grade of C– in all of the following classes or their equivalents. *None of these classes may be taken on a credit/no-credit basis.*
 - Computer Science 2010, 2020, and 2100.
 - Mathematics 1210 and 1220.
 - Physics 2210.
 - University English writing requirement
2. A grade of CR in CS 1010 (a credit/no-credit class).
3. A cumulative grade point average of 2.25 or higher. (Note that *much* higher grades in the required classes listed above are typically needed. See below for details.)

A student may apply for full major status no earlier than the semester during which he or she expects to complete these requirements. Applications may be submitted any time during the first session of each semester. Students whose applications are accepted will be advanced to full major status effective the following semester. (For example, students who apply between August 22 and October 15, 2001, and who are subsequently accepted, will become full majors at the beginning of the spring 2002 semester.) Students whose applications are not accepted are free to reapply during subsequent semesters.

The School determines how many new majors will be admitted each semester based on laboratory facilities, computer resources, and available faculty. The following factors are considered in determining which students to admit:

1. GPA in the required pre-major classes listed above.
2. A personal goals statement.
3. ACT or SAT scores.
4. An optional programming portfolio. (This should be submitted only by students who have written significant, interesting programs outside of a classroom setting.)

No pre-major class may be taken more than twice. If a class is repeated, the grade received the second time is used. If a student receives *any* grade in a class—including W (withdrawal), I (incomplete), or V (audit)—the student is considered to have taken the class. Only three classes may be repeated without penalty. For any additional classes that a student repeats, only 80% of the grade points received in the repeated class will be used in the GPA calculation.

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

Each student who majors in computer science is assigned a faculty advisor. The School also has an Undergraduate Academic Counselor (Sandy Hiskey, 3190 MEB, 581-8224, shiskey@cs.utah.edu) who is available to answer questions regarding registration for CS classes, transfer of credits, degree requirements, recent School actions, etc. The Academic Counselor can also arrange appointments with the faculty advisor.

A student is welcome to meet with the faculty advisor whenever necessary to discuss schedule plans or current problems. The responsibility for arranging an appointment is left to the student. Students should always feel free to seek advice from the advisor regarding their programs and plans.

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 an intensive set of computer science, calculus, and physics courses during the freshman year. Only strong training in high school will allow a student to begin at this level. 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 and 1220, 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 the pre-major requirements.) *These six courses must include one pair of courses that form an approved concentration, one upper division course, and either one additional concentration or one additional upper division course.*

In a concentration, the second course further develops ideas or issues introduced in the first course. A list of sample concentrations and the General Education Program 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 2740, History 1700, Honors 2212, or Political Science 1100.

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 Writing 3400, which is required for computer science majors.
- (b) The quantitatively intensive course requirement is satisfied by CS 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. Math, Science, and Engineering: Seven classes in math, science, and/or engineering are required.

- (a) Mathematics 1210 and 1220 (Calculus I and II) are required.
- (b) Physics 2210 (Physics I) is required.
- (c) Students must take four additional courses, *each of which must be at least three semester hours*, chosen from the following list. Students should take the prerequisites of computer science electives into consideration when planning how to satisfy this requirement.
- i. Physics 2220 (Physics II)
 - ii. Biology 1000 (General Biology)
 - iii. Chemistry 1210 (General Chemistry)
 - iv. 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.

4. 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:

CS 1010 Introduction to Unix
 CS 2010 Introduction to Computer Science I
 CS 2020 Introduction to Computer Science II
 CS 2100 Discrete Structures
 CS 3500 Software Practice
 CS 3510 Algorithms and Data Structures
 CS 3810 Computer Architecture
 CS 4400 Computer Systems
 CS 4500 Software Engineering Lab

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

CS 3100 Models of Computation
 CS 3200 Scientific Computing

| Year | Full major status after two semesters | | | | Full major status after three semesters | | | |
|------|---------------------------------------|---------------|---------------------------|-------------|---|---------------|---------------------------|-------------|
| | Fall | | Spring | | Fall | | Spring | |
| 1 | Math 1210 [†] | (4) | Math 1220 [†] | (4) | Math 1210 [†] | (4) | Math 1220 [†] | (4) |
| | CS 1010 [†] | (0.5) | Physics 2210 [†] | (4) | CS 1010 [†] | (0.5) | Physics 2210 [†] | (4) |
| | CS 2010 [†] | (4) | CS 2020 [†] | (4) | Writing [†] | (3) | CS 2010 [†] | (4) |
| | Writing [†] | (3) | CS 2100 [†] | (2) | Gen Ed | (3) | Gen Ed | (3) |
| | Gen Ed | (3) | | | Gen Ed | (3) | | |
| | | <u>(14.5)</u> | | <u>(14)</u> | | <u>(13.5)</u> | | <u>(15)</u> |
| 2 | CS 3500 | (4) | CS 3510 | (3) | CS 2020 [†] | (2) | CS 3510 | (3) |
| | CS 3810 | (4) | CS 4400 | (3) | CS 2100 [†] | (4) | CS theory elective | (3) |
| | Writing 3400 | (3) | Math/sci elective | (3) | Math/sci elective | (3) | Math/sci elective | (4) |
| | Gen Ed | (4) | Gen Ed | (3) | Gen Ed | (3) | Writing 3400 | (3) |
| | | | Free elective | (3) | Gen Ed | (3) | Free elective | (3) |
| | | <u>(15)</u> | | <u>(15)</u> | | <u>(15)</u> | | <u>(16)</u> |
| 3 | CS theory elective | (3) | CS elective | (3) | CS 3500 | (4) | CS 4400 | (3) |
| | CS elective | (3) | CS elective | (3) | CS 3810 | (4) | CS elective | (3) |
| | Math/sci elective | (4) | Math/sci elective | (4) | Math/sci elective | (4) | CS elective | (3) |
| | Gen Ed | (3) | Gen Ed | (3) | Gen Ed | (4) | Math/sci elective | (4) |
| | Free elective | (3) | Free elective | (3) | | | Free elective | (3) |
| | | <u>(16)</u> | | <u>(16)</u> | | <u>(16)</u> | | <u>(16)</u> |
| 4 | CS elective | (3) | CS 4500 | (3) | CS elective | (3) | CS 4500 | (3) |
| | CS elective | (3) | CS elective | (3) | CS elective | (3) | CS elective | (3) |
| | CS elective | (3) | Math/sci elective | (4) | CS elective | (3) | CS elective | (3) |
| | Gen Ed | (3) | Gen Ed | (3) | Gen Ed | (4) | Free elective | (3) |
| | Free elective | (4) | Free elective | (3) | Free elective | (3) | Free elective | (3) |
| | | <u>(16)</u> | | <u>(16)</u> | | <u>(16)</u> | | <u>(15)</u> |

These tables give two example eight-semester programs leading to a Bachelor of Science degree in computer science. The program on the left is for students who advance to full major status after two semesters, and the program on the right is for students who advance to full major status after three semesters (not counting summers). It is meant only as a guide, since the scheduling of electives and General Education classes depends on which ones are selected. Advancement to full major status in two semesters requires taking Computer Science 2010, Math 1210, and Writing during the first semester, unless the student has advanced placement or plans to take courses during the summer. ([†]Class required of pre-majors.)

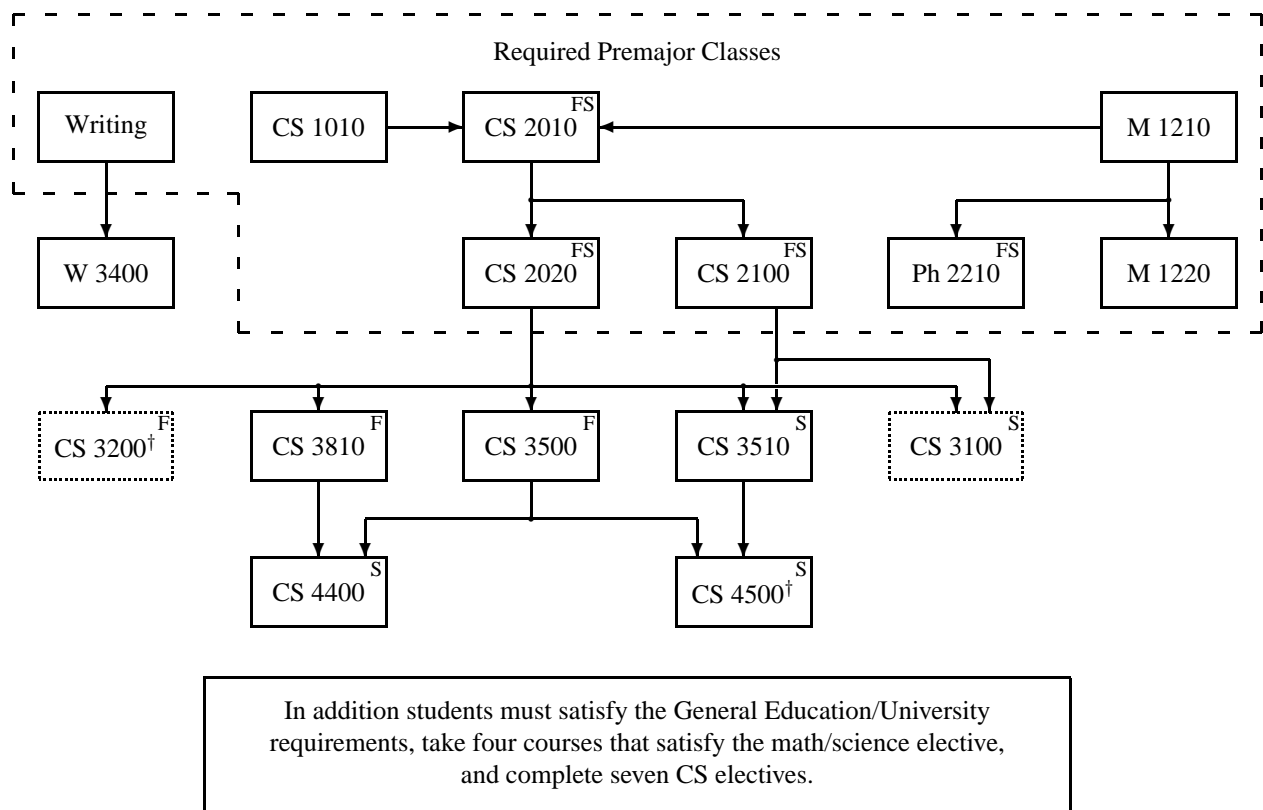
Figure 1.1: Example Computer Science Degree Program

(c) Electives. Seven additional Computer Science classes numbered at the 3000, 4000, or 5000 level, totaling at least 21 semester hours, must be taken. CS 3050/51, 5010/20, 5050/51, seminars, and independent study may not be counted. Only one of CS 3610 and CS 5600 may be counted.

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

5. **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 CS 1010, in which a grade of CR is required). A student may repeat such courses only one time. With the exception of the American Institutions requirement and CS 1010, all courses listed above must be taken for a letter grade (i.e., they may not be taken CR/NC).*

In order to remain in good standing and to graduate, a student is required to maintain a cumulative grade point average at the University of 2.25 or higher, and also to maintain a grade point average of 2.25 in computer science classes. Students whose grade point average in either of these two categories falls below 2.25 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.



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. (†CS 3200 has Math 2250 as a prerequisite; CS 4500 can be taken only by students who are on track to graduate before the next offering of CS 4500.)

Figure 1.2: Computer Science Prerequisites

All students admitted as full majors are placed on probationary status. If a student's GPA in either of the above categories is below 2.25 at the end of the first academic year during which they take upper level CS 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 CS 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.

1.4 Computer Science Honors Degrees

The School faculty has voted to offer two different Honors Bachelor's Degrees in Computer Science. (One degree will have an emphasis similar to the current Bachelor's Degree program, and the other will have an emphasis in computer graphics.)

As of June 2001, however, the degrees had not yet been approved by the University administration. Because changes in the proposed requirements will likely be made during the University approval process, we have not included the honors degree requirements in this edition of the handbook. We anticipate, however, that we will obtain approval of the two honors programs in time for the fall semester of 2001. At that time, an updated handbook will be issued.

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.

A limited number of students are admitted into the minor program each year. In order to be admitted, a student must have a declared major in another department and be making progress in that major. The maximum number of students accepted into the minor is determined by the faculty on an annual basis and is currently about 10.

The admission process is similar to that for majors and is carried out at the same times. Students compete for admission based on grades in Math 1210, English writing, and Computer Science 2010/2020/2100.

The minor consists of a minimum of 17.5 semester hours of computer science classes.

- Required. The following classes must be taken.

| | |
|--------------|----------------------------------|
| CS 1010 | Introduction to Unix |
| CS 2010/2020 | Introduction to Computer Science |
| CS 2100 | Discrete Structures |
| CS 3500 | Software Practice |

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

Computer science minors are guaranteed admission into only the upper division classes that comprise their minor. They are not allowed to take computer science classes numbered 4000 or above. Students in the minor will not be able to do phone registration for upper division classes; they must add them through the undergraduate office each semester.

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 or mathematics, where upper level classes may serve as computer science electives. Students pursuing a double major must notify the Academic Counselor.

1.7 Course Enrollment Priorities for Non-Majors

All of the upper division computer science courses are listed with the Registrar as available only to full majors in computer science or computer engineering. This means that other students, including computer science minors, will not be able to pre-register for said classes, and must place their names on a waiting list maintained in the Undergraduate Office prior to the first day of class. If there are vacancies in the classes, the School's director will decide who will receive permission to add the classes based on considerations such as GPA and grades in prerequisites. Computer science and computer engineering majors, and computer science minors, will receive priority in classes needed to complete their program.

1.8 Undergraduate Financial Assistance

The School of Computing has several financial assistance awards available. Applications may be picked up from the Academic Counselor in the School's administrative office. These scholarships are awarded based upon academic performance, rather than financial need. Most are awarded to CS/CE full majors or those students who will become full majors during the following academic year. The following scholarships are available. All applications must be submitted to the School's office by March 15.

Special Departmental Scholarship. These awards are available to students majoring in Computer Science and 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 majors. These are unrestricted cash awards of \$500 to \$2,500, made possible in part by generous donations by Novell and the Eccles Foundation. To be eligible, students must take at least 8 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 or in Computer Engineering. These awards are unrestricted cash awards of \$500 to \$2000. To be eligible, students must take at least 8 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 upon need. Obtain an application form from the Office of the Dean of Engineering in Kennecott 214. Applications must be submitted by February 15.

1.9 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 sometime in the future, and valuable experience in an area that is pertinent to current studies. Among the corporations participating are 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 in the Undergraduate Office. 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 posted on a computer bulletin board which is accessible to 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.

1.10 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) and the Student Chapter of the Association of Computing Machinery (ACM). UGSAC and ACM play an active role in the School and coordinate 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 in February.
6. Organization of lunch meetings for pre-majors and majors.
7. Organization of university and high school programming contests.
8. Feedback on School issues affecting students, such as scheduling, curriculum changes, and graduation requirements.
9. Course evaluations.
10. GRE preparation classes.

Anyone interested in joining either of these 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 an ATM fabric running at OC-12 (622 Mbps) and that provides desktop connections at speeds ranging from Fast Ethernet (100 Mbps) 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 Solaris-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 over 180 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 NT Lab in EMCB 210 includes approximately 90 Pentium II-based PCs. The electronic classroom in MEB 3225 contains 30 Pentium III-based PCs arranged into a classroom configuration. The CES/Grad Lab in MEB 3161 contains nine SGI workstations. 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.

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 undergraduate and the 6000-level version 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.

Current class schedules and registration information¹ are available on line.

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

An introduction to the use of widely available software tools to solve computational problems in science and engineering. Introductions to numerical and symbolic programming using Maple, and to procedural programming using C. Elementary numerical methods. Emphasis on the entire process of solving problems from science and engineering via computational means.

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.

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.

¹<http://www.acs.utah.edu/prod/bin/student>

1050 Social Aspects of a Digital World (2, F)

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 1000-level discussion course (such as CP SC 1051) is required. (Not offered 2001–02.)

1051 Introductory Discussion of Social Aspects (1, F)

The combination of CP SC 1050/1051 is appropriate for any student who is already comfortable using a computer to write papers and explore the Web. (Not offered 2001–02.)

1950 Independent Study (1 to 4)**1960 Special Topics (1 to 4)****2000 Introduction to Program Design in C++ (4, F) Coreq.: MATH 1210, CP SC 1010**

Introduction to essential programming concepts using C++. Decomposition of programs into functional units, control structures, fundamental data structures of C++, dynamic memory management, data representation, low-level programming. Laboratory practice. (Intended for non-CS/CE majors; students planning to take CS 2020 should take CS 2010.)

2010 Introduction to Computer Science I (4, FS) Coreq.: MATH 1210, CP SC 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.: CP SC 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 (2, FS) Prereq.: CP SC 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)****3050 Social Aspects of a Digital World (2, F) Prereq.: Programming proficiency**

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 3000-level discussion course (such as CP SC 3051) is required. (Not offered 2001–02.)

3051 Intermediate Discussion of Social Aspects (1, F) Prereq.: Programming proficiency

The combination of CP SC 3050/3051 is appropriate for students who have an understanding of computing technology comparable to that of a newly-admitted computer science major. (Not offered 2001–02.)

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

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

3200 Scientific Computation (3, F) Prereq.: CP SC 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).

3500 Software Practice (4, F) Prereq.: CP SC 2020

Meets with CP SC 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 (3, S) Quantitatively Intensive B.S. Course. Prereq.: CP SC 2020, CP SC 2100

Meets with CP SC 5020. Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CP SC 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.

3520 Programming Language Concepts (3, F) Prereq.: CP SC 3500, CP SC 3510

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.

3600 Fundamentals of Computer Graphics I (4, F) Prereq.: Admission to the Computer Graphics and Visualization honors track; Coreq.: CP SC 2020

Applied geometry, linear algebra, basic numerical techniques. Software design, implementation, and testing. Basic raster graphics, 2D and 3D transforms, hidden-surface elimination.

3610 Fundamentals of Computer Graphics II (4, S) Prereq.: CP SC 3600

Shading models, geometric modeling, physically-based animation, color theory.

3700 Fundamentals of Digital System Design (4, S) Quantitatively Intensive B.S. Course. Cross-listed as EL EN 3700. Prereq.: CP SC 2010, PHYCS 2220

Techniques for minimizing logic functions and designing common combinational circuits such as decoders, selectors, and adders. Synchronous and asynchronous sequential circuits, state diagrams, Mealy and Moore circuits, state minimization and assignment. Use of software tools for design, minimization, simulation, and schematic capture. Implementation with MSI, LSI, and field programmable gate arrays. Laboratory included.

3710 Computer Design Laboratory (3, F) Cross-listed as EL EN 3710. Prereq.: CP SC/EL EN 3700, CP SC/EL EN 3810

Student groups design, build, and test a programmable device such as a computer or calculator.

3720 Analog & Digital Interfacing with Microprocessors & Microcontrollers (4, S) Cross-listed as EL EN 3720. Prereq.: CP SC/EL EN 3700

Fundamentals of digital-to-analog (D-to-A) and analog-to-digital (A-to-D) circuits, relays, stepper motors, and digital switches. Interfacing digital and analog circuits to computers and micro-controllers. Laboratory included.

3810 Computer Architecture (4, F) Quantitatively Intensive B.S. Course. Cross-listed as EL EN 3810. Prereq.: CP SC 2020

An in-depth study of computer architecture and design, from digital logic to operating systems, including topics such as pipelining, memory systems, parallel and serial communication, and interrupts. Performance measures and compilation issues. Computer architectures including RISC, CISC, stack, and parallel.

3950 Independent Study (1–4)

3960 Special Topics (1–4)

4400 Computer Systems (3, S) Prereq.: CP SC 3500, CP SC 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.

4500 Software Engineering Laboratory (3, S) Prereq.: CP SC 3500, CP SC 3510, senior standing in Computer Science

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

4550 Simulation (3) Prereq.: CP SC 3500, CP SC 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. (Not offered 2001–02.)

4710 Computer Engineering Senior Project (3, F) Cross-listed as EL EN 4710. Prereq.: CP SC/EL EN 3710, CP SC/EL EN 3720, senior standing in Computer Engineering

Students design a microcomputer system that includes RAM, EPROM, and I/O devices. Capstone project for computer engineering majors. Formal written reports, one or more oral presentations.

4950 Independent Study (1–4)

4960–4964 Special Topics (1–4)

The following special topics courses are currently scheduled for the 2001–2002 academic year. Contact the instructor for details.

- **CP SC 4960 Teaching Introductory Computer Science** (1, FS). Prof. Zachary.

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.

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, F) Prereq.: CP SC 2020 and permission of instructor

Meets with CP SC 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, S) Prereq.: CP SC 5010 and permission of instructor

Meets with CP SC 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 CP SC 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.

5050 Social Aspects of a Digital World (2, F) Prereq.: Permission of instructor

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 5000-level discussion course (such as CP SC 5051) is required. (Companion discussions may also be offered by other departments.) (Not offered 2001–02.)

5051 Advanced Discussion of Social Aspects (1, F) Prereq.: Permission of instructor

The combination of CP SC 5050/5051 is appropriate for students who have a graduate-level background in issues related to the social aspects of computing (e.g., intellectual property or electronic commerce). (Not offered 2001–02.)

5060 Legal Protection of Digital Information (2)

Ways of protecting digital information—computer software and databases—using intellectual property law. Copyrights, patents, trade secrets, and contracts as ways of protecting digital information. (Not offered 2001–02.)

5100 Foundations of Computer Science (3, F) Prereq.: CP SC 3100, CP SC 3500, CP SC 3510

Meets with CP SC 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.: CP SC 3200, CP SC 3500, CP SC 3510, MATH 3160

Meets with CP SC 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, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 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.: CP SC 1000, MATH 2250, PHYCS 2220

Meets with CP SC 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.: CP SC 3500, CP SC 3510, MATH 2210, MATH 2270

Meets with CP SC 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.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 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, S) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 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.

5460 Operating Systems (3, F) Prereq.: CP SC 3510, CP SC/EL EN 3810, CP SC 4400

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 (3, S) Prereq.: CP SC 3100, CP SC 3510, CP SC/EL EN 3810, CP SC 4400

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 Data Communications and Networks (3, F) Prereq.: CP SC 3510, CP SC/EL EN 3810, CP SC 4400

Meets with CP SC 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.

5530 Database Systems (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 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.: CP SC 3500, CP SC 3510

Meets with CP SC 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.: CP SC 3500, MATH 2250; Coreq.: CP SC 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.

5610 Computer Graphics I (3, F) Prereq.: CP SC 5600 or CP SC 3610

Meets with CP SC 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.: CP SC 3500, CP SC 3510; CP SC 3200 or CP SC 5210 or MATH 5600

Meets with CP SC 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.

5710 Advanced Integrated Circuit Design I (3, F) Cross-listed as EL EN 5710. Prereq.: CP SC/EL EN 3700

Meets with CP SC 6710. Introduction to basic concepts of the design of CMOS integrated circuits for students with a wide range of backgrounds. Static and dynamic properties of CMOS circuits, composite layout of CMOS circuits, and modeling of transistors for use in SPICE simulations. Commonly encountered CMOS circuits. Introduction to CMOS analog/digital circuits. Students complete design, composite layout, and digitization of a simple integrated circuit using computer-aided design tools.

5720 Advanced Integrated Circuit Design II (3) Cross-listed as EL EN 5720. Prereq.: CP SC/EL EN 5710/6710, EL EN 2100

Meets with CP SC 6720. Design of mixed signal (analog/digital) CMOS integrated circuits. Fundamental building blocks for analog circuits, including the basic principles of opamp, current mirror and comparator design. Basics of discrete-time signals and filters. Implementation of switched capacitor circuits and discussions of various implementations of D/A and A/D converters, oversampled converters and phase locked loops. (Not offered 2001–02.)

5740 Computer-Aided Design of Digital Circuits (3) Cross-listed as EL EN 5740. Prereq.: CP SC/EL EN 3700, CP SC 3510

Meets with CP SC 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. (Not offered 2001–02.)

5750 Synthesis and Verification of Asynchronous VLSI Systems (3, F) Cross-listed as EL EN 5750. Prereq.: CP SC/EL EN 3700, CP SC 3510

Meets with CP SC 6750. Introduction to systematic methods for the design of asynchronous VLSI systems from high-level specifications to efficient, reliable circuit implementations. Topics include specification, controller synthesis, optimization using timing information, technology mapping, data path design, and verification.

5810 Advanced Computer Architecture (3, F) Cross-listed as EL EN 5810. Prereq.: CP SC/EL EN 3700, CP SC/EL EN 3810

Meets with CP SC 6810. 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.

5830 VLSI Architecture (3) Cross-listed as EL EN 5830. Prereq.: CP SC/EL EN 3700, CP SC/EL EN 3810

Meets with CP SC 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. (Not offered 2001-02.)

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 2001–02 academic year. Contact the faculty member in charge for details.

- **CP SC 5960 Programming Language Design - Java** (3,S). Prof. Lindstrom.
- **CP SC 5963 Advanced Manufacturing** (Arr,F). Prof. Drake.

6010 Writing Research Proposals (2) 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. Not offered 2001–02.)

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 CP SC 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 CP SC 6020 within three years of enrolling in the graduate program. CP SC 6020 may not be repeated for credit.

6100 Foundations of Computer Science (3, F) Prereq.: CP SC 3100, CP SC 3500, CP SC 3510

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

6110 Formal Methods for System Design (3, S) Prereq.: CP SC 5100/6100 and CP SC 6520

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.: CP SC 3200, CP SC 3500, CP SC 3510, MATH 3160

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

6220 Advanced Scientific Computing II (3, S) Prereq.: CP SC 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, F) Prereq.: CP SC 3500, CP SC 3510

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

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

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

6320 Computer Vision (3, S) Prereq.: CP SC 3500, CP SC 3510, MATH 2210, MATH 2270

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

6340 Natural Language Processing (3, F) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

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

6350 Machine Learning (3, S) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

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

6360 Virtual Reality (3, S) Prereq.: CP SC 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. (Not offered 2001–02.)

6470 Advanced Topics in Compilation (3, F) Prereq.: CP SC 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.: CP SC 3500, CP SC 3510, CP SC/EL EN 3810

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

6520 Programming Languages and Semantics (3, S) Prereq.: CP SC 3520, CP SC 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.

6530 Database Systems (3, F) Prereq.: CP SC 3500, CP SC 3510

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

6540 Human/Computer Interaction (3, F) Prereq.: CP SC 3500, CP SC 3510

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

6610 Computer Graphics I (3, S) Prereq.: CP SC 5600 or CP SC 3610

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

6620 Computer Graphics II (3, S) Prereq.: CP SC 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.: CP SC 3500, CP SC 3510; CP SC 3200 or CP SC 5210/6210 or MATH 5600

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

6650 Image Synthesis (3, F) Prereq.: CP SC 5620/6620, CP SC 6670, MATH 5010

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

6670 Computer-Aided Geometric Design I (3, F) Prereq.: MATH 2210, MATH 2250, CP SC 3500, CP SC 3510; Coreq.: CP SC 5600/6600**6680 Computer-Aided Geometric Design II** (3) Prereq.: CP SC 6670

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. (Not offered 2001–02.)

6710 Advanced Integrated Circuit Design I (3, F) Cross-listed as EL EN 6710. Prereq.: CP SC/EL EN 3700

Meets with CP SC 5710. Graduate and honors students only. Extra work required.

6720 Advanced Integrated Circuit Design II (3) Cross-listed as EL EN 6720. Prereq.: CP SC/EL EN 5710/6710, EL EN 2100

Meets with CP SC 5720. Graduate and honors students only. Extra work required. (Not offered 2001–02.)

6740 Computer-Aided Design of Digital Circuits (3) Cross-listed as EL EN 6740. Prereq.: CP SC/EL EN 3700, CP SC 3510

Meets with CP SC 5740. Graduate and honors students only. Extra work required. (Not offered 2001–02.)

6750 Synthesis and Verification of Asynchronous VLSI Systems (3, F) Cross-listed as EL EN 6750. Prereq.: CP SC/EL EN 3700, CP SC 3510

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

6770 Advanced Digital VLSI Systems Design (3) Cross-listed as EL EN 6770. Prereq.: CP SC/EL EN 5710/6710

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. (Not offered 2001–02.)

6810 Advanced Computer Architecture (3, F) Cross-listed as EL EN 6810. Prereq.: CP SC/EL EN 3700, CP SC/EL EN 3810

Meets with CP SC 5810. Graduate and honors students only. Extra work required.

6820 Parallel Computer Architecture (3) Cross-listed as EL EN 6820. Prereq.: CP SC/EL EN 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. (Not offered 2001–02.)

6830 VLSI Architecture (3) Cross-listed as EL EN 6830. Prereq.: CP SC/EL EN 3700, CP SC/EL EN 3810

Meets with CP SC 5830. Graduate and honors students only. Extra work required. (Not offered 2001–02.)

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 2001–2002 academic year. Contact the instructor for details.

- **CP SC 69xx Signal Processing** (3,S). Prof. Kowalski.

6970 Masters Thesis Research (1–12)

6980 Faculty Consultation Masters (1–12)

7120 Information-Based Complexity (3) Prereq.: CP SC 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. (Not offered 2001–02.)

7240 Sinc Methods (3, S) Prereq.: CP SC 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.: CP SC/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.

7460 Advanced Operating Systems (3) Prereq.: CP SC 5460, CP SC 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. (Not offered 2001–02.)

7940 Seminar (1–3)

May be repeated for credit.

7950 Independent Study (1–4)**7960 Special Topics** (1–4)**7970 PhD Dissertation Research** (1–12)**7980 Faculty Consultation PhD** (1–12)**7990 Continuing Registration: PhD** (0)

4

School of Computing Faculty and Their Research Interests

Erik Brunvand

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

Professor Brunvand¹ 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 Carter² 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.

¹<http://www.cs.utah.edu/~elb/>

²<http://www.cs.utah.edu/~retrac/>

Elaine Cohen

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

Professor Cohen³ 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.

Al Davis

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

Professor Davis⁴ 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's⁵ 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.

³<http://www.cs.utah.edu/~cohen/>

⁴<http://www.cs.utah.edu/~ald/>

⁵<http://www.cs.utah.edu/~mflatt/>

Ganesh C. Gopalakrishnan

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

Professor Gopalakrishnan's⁶ 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.

David H. Hanscom

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

Professor Hanscom's⁷ 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 Hansen⁸ 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's⁹ 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.

⁶<http://www.cs.utah.edu/~ganesh/>

⁷<http://www.cs.utah.edu/~hanscom/>

⁸<http://www.cs.utah.edu/~hansen/>

⁹<http://www.cs.utah.edu/~tch/>

Lee A. Hollaar

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

Professor Hollaar's¹⁰ 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.

John M. Hollerbach

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

Professor Hollerbach's¹¹ 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. In robotics, autonomous and teleoperated grasping is pursued with a pair of left/right Utah/MIT Dextrous Hands. Teleoperation research is focusing on the use of the Sarcos Dextrous Arm Master and Slave, an advanced hydraulic telemanipulator.

Wilson C. Hsieh

Assistant Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1995

Professor Hsieh¹² joined the faculty in September 1997. His research interests are in compilers and programming languages, operating systems, and architecture.

Christopher R. Johnson

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

Professor Johnson's¹³ 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.

¹⁰<http://www.cs.utah.edu/~hollaar/>

¹¹<http://www.cs.utah.edu/~jmh/>

¹²<http://www.cs.utah.edu/~wilson/>

¹³<http://www.cs.utah.edu/~crj/>

Robert Kessler

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

Professor Kessler's¹⁴ 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.

Arthur H. Lee

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

Professor Lee¹⁵ 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 Associate Professor, School of Computing
B.S., University of Utah, 1983

Professor Lepreau's¹⁶ 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's¹⁷ 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.

¹⁴<http://www.cs.utah.edu/~kessler/>

¹⁵<http://www.cs.utah.edu/~alee/>

¹⁶<http://www.cs.utah.edu/~lepreau/>

¹⁷<http://www.cs.utah.edu/~gary/>

Sally McKee

Research Assistant Professor, School of Computing
Ph.D., University of Virginia, 1995

Professor McKee¹⁸ joined the faculty in 1998. Her research interests lie in the intersection of computer architecture, operating systems, compilers, and performance evaluation. Her current work focuses on memory system design and analysis. Dr. McKee is a member of the Impulse Adaptable Memory Systems project, where she is helping to build a computer system that gives applications more control over how their data is accessed. Another project investigates how to give the memory hierarchy knowledge of future access patterns, and then how best to exploit that knowledge to improve system performance. A third project seeks to develop an analytical method for designing and evaluating better complex cache hierarchies.

Richard F. Riesenfeld

Professor, School of Computing
Ph.D., Syracuse University, 1973

Professor Riesenfeld¹⁹ 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 Riloff²⁰ 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.

Peter Shirley

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

Professor Shirley²¹ 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.

¹⁸<http://www.cs.utah.edu/~sam/>

¹⁹<http://www.cs.utah.edu/~rfr/>

²⁰<http://www.cs.utah.edu/~riloff/>

²¹<http://www.cs.utah.edu/~shirley/>

Kris Sikorski

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

Professor Sikorski's²² 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.

Konrad Slind

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

Professor Slind²³ 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.

Kent F. Smith

Professor, School of Computing
Professor of Electrical Engineering
Ph.D., University of Utah, 1982

Professor Smith's²⁴ principal interests lie in the design of integrated circuits using computer aids for structured logic. His present research is focused on techniques for high speed GaAs integrated circuits. This research involves the detailed design and fabrication of the actual integrated circuits as well as Computer Aided Design (CAD) tools for the design of these circuits. The GaAs circuits he is working with operate at speed in excess of 10 GHz and thus present problems involving very difficult analysis. For example, high speed design requires that the components be physically placed near each other to give optimum performance. In addition, the actual interconnecting wires between logic gates must be characterized and used in the analysis. The extraction and use of these high speed parameters represents problems that require new methods of design. Prior to joining the University of Utah faculty, Professor Smith was responsible for integrated circuit design and testing at the Microcircuit Laboratory at the University of Utah Research Institute. Prior to that time he was the technical director for Electrical Engineering at the General Instrument Advanced Microelectronics Lab. He holds a number of patents in circuits primarily concerning MOS and bipolar integrated circuits.

Frank Stenger

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

Professor Stenger's²⁵ 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

²²<http://www.cs.utah.edu/~sikorski/>

²³<http://www.cs.utah.edu/~slind/>

²⁴<http://www.cs.utah.edu/~k-smith/>

²⁵<http://www.cs.utah.edu/~stenger/>

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 Thompson²⁶ 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's²⁷ 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

Assistant Professor, School of Computing
Ph.D., University of North Carolina, 1993

Professor Whitaker²⁸ 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.

Joseph L. Zachary

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

Professor Zachary²⁹ 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.

²⁶<http://www.cs.utah.edu/~cindi/>

²⁷<http://www.cs.utah.edu/~thompson/>

²⁸<http://www.cs.utah.edu/~whitaker/>

²⁹<http://www.cs.utah.edu/~zachary/>