

Computer Science Undergraduate Handbook

**University of Utah
Department of Computer Science
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

1998–1999

The Department of Computer Science offers a Bachelor of Science degree in Computer Science. The undergraduate program begins with a two-course sequence that gives students solid programming skills while exposing them to the breadth of issues that arise in computer science. Students then take four core courses in software engineering, computer architecture, algorithms and data structures, and formal models of computation. Students build on this background by choosing four additional core courses (from among operating systems, programming languages, compilers, numerical analysis, and digital design) and by choosing four electives from the breadth of the department'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 either a senior project or thesis. Along with an in-depth study of computing, the undergraduate curriculum encompasses a general education in mathematics, science, and the humanities.

The department also offers a minor in Computer Science for students who want to use computers in another field or to teach computer science at the secondary school level. 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 Departments of Computer Science and Electrical Engineering. Information about that program is in a separate handbook that can be obtained from either department's office or from the web page referenced below. The Department of Computer Science is currently responsible for counseling Computer Engineering majors.

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/csinfo/handbooks>.)

Faculty

| | | |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Professors | Elaine Cohen, Ph.D. Thomas C. Henderson, Ph.D. John M. Hollerbach, Ph.D. Robert Kessler, Ph.D. Kent F. Smith, Ph.D. William B. Thompson, Ph.D. | Alan L. Davis, Ph.D. Lee A. Hollaar, Ph.D. Gary E. Lindstrom, Ph.D. Richard F. Riesenfeld, Ph.D. Frank Stenger, Ph.D. |
| Emeritus Professor | Robert R. Johnson, Ph.D. | |
| Associate Professors | Erik Brunvand, Ph.D. Christopher R. Johnson, Ph.D. | Ganesh Gopalakrishnan, Ph.D. Kris Sikorski, Ph.D. |
| Assistant Professors | John Carter, Ph.D. Ellen M. Riloff, Ph.D. | Wilson Hsieh, Ph.D. Peter Shirley, Ph.D. |
| Clinical Professor | David Hanscom, Ph.D. | |
| Clinical Associate Professor | Joseph L. Zachary, Ph.D. | |
| Research Professor | Stephen Jacobsen, Ph.D. | |
| Research Associate Professors | Tony Carter, Ph.D. Chuck Hansen, Ph.D. | Sam Drake, Sc.D. |
| Research Assistant Professors | Jay Lepreau Mark Swanson, Ph.D. | Chris Myers, Ph.D. |
| Adjunct Professors | Andries van Dam, Ph.D. Peter Kind, Lt. Gen., US Army (Ret.) | Martin Griss, Ph.D. |
| Adjunct Associate Professors | Robert J. Douglass, Ph.D. | Robert McDermott, Ph.D. |
| Adjunct Assistant Professor | Don Brown, Ph.D. | |
| Visiting Associate Professor | Abdel Mokkedem, Ph.D. | |

Administration

| | | |
|------------------------|-------------------------------------------------|----------|
| Robert R. Kessler | Chair | 581-4653 |
| John Hollerbach | Associate Chair, Research | 585-6978 |
| Joseph L. Zachary | Associate Chair, Academics | 581-7079 |
| Erik Brunvand | Director, Graduate Studies | 581-4345 |
| David Hanscom | Director, Computer Engineering | 581-7023 |
| | Director, Undergraduate Studies | |
| Christopher R. Johnson | Director, Computational Engineering and Science | 581-7705 |
| Joseph L. Zachary | Director, Educational Programs | 581-7079 |

Staff

| | | |
|-----------------|-------------------------------------------|----------|
| John Allen-Rose | Computer Professional | 581-8065 |
| Steve Backus | Computer Professional | 581-5844 |
| Mark Bradakis | Computer Professional | 585-3358 |
| Erin Davies | Secretary (part-time) | 581-8224 |
| Ken Davis | Associate Accountant | 585-5025 |
| Monica Heaton | Administrative Assistant | 581-8224 |
| Shawn Darby | Administrative Manager II | 581-3950 |
| Todd Green | Computer Administrator | 581-3569 |
| Corey Hatch | Supervisor, Digital Systems Laboratory | 581-7845 |
| Colleen Hoopes | Graduate Academic Program Specialist | 581-8224 |
| Chad Lake | Computer Professional | 581-8224 |
| Bill Martens | Computer Professional | 581-8065 |
| Marti Porter | Secretary (part-time) | 581-8224 |
| Raelynn Potts | Executive Secretary | 585-5983 |
| Mary Rawlinson | Undergraduate Academic Program Specialist | 581-8224 |
| John Smith | Computer Professional | 581-8713 |
| Woody Walton | Facility Staff Research Assistant | 581-8065 |

Contents

| | |
|-------------------------------------------------------------------|------------|
| Faculty | iii |
| Administration | v |
| 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 B.S. Degree | 3 |
| 1.4 Computer Science as a Minor | 6 |
| 1.5 Students Pursuing a Second Degree or a Double Major | 7 |
| 1.6 Course Enrollment Priorities for Non-Majors | 7 |
| 1.7 Undergraduate Financial Assistance | 7 |
| 1.8 Employment Opportunities | 8 |
| 1.9 Student Participation in Departmental Affairs | 8 |
| 2 Departmental Facilities | 9 |
| 3 Computer Science Courses | 11 |
| 4 CS Faculty And Their Research Interests | 19 |

1

The Computer Science Major and Minor

The Department of Computer Science offers a Bachelor of Science (B.S.) degree in Computer Science. This is a software-oriented degree whose requirements include 16 courses from the department. A student must be admitted as a major by the department in order to take upper-division courses and pursue the Computer Science degree. The department also offers a minor in Computer Science consisting of 6 courses from the department. Students from the College of Education can use this minor as part of their teaching minor.

Because of the transition to semesters during the 1998–99 academic year, the requirements for students applying for full major status during the summer of 1998 are slightly different than those for students applying during the summer of 1999 or later. The requirements listed below are as specific as possible in that regard. If special cases arise, consult with the Academic Counselor.

The requirements in this chapter do not apply to students who were admitted to full major status prior to the summer of 1998. Such students should consult previous years' handbooks and complete an Academic Program Completion Plan.

Throughout this handbook, courses with three-digit numbers are quarter-length courses offered prior to the fall semester of 1998, and courses with four-digit numbers are semester-length courses offered thereafter.

1.1 Becoming a Computer Science Major

Any student may become a Computer Science pre-major by filling out the appropriate form provided by the Registrar. It is advisable to do this early to ensure receiving departmental information and staying advised of any changes that may be made in degree requirements. Declaration of a major will also enable participation in department affairs such as the Student Advisory Committee (SAC).

In order to become a full major, a student must complete the courses required of pre-majors and apply for full major status. An application should be filled out at the department office before the end of the spring semester of the year of application, whether or not the student intends to take additional required classes during the summer. A student may not pre-register for any upper-division (3000 or higher) classes in Computer Science without first being admitted to full major status. Applications for admission are reviewed between May and August. The department determines how many new majors will be admitted each year, based on laboratory facilities, computer resources, and available faculty.

To be considered for admission to full major status during the summer of 1998, a student must have:

1. A minimum grade of C– in all of the following classes or their equivalents:
 - Mathematics 111/112/113
 - Physics 221/222
 - Computer Science 201/202
 - University English writing requirement
2. A grade of CR in CS 110 (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 required. See below for details.)

To be considered for admission to full major status during the summer of 1999 or later, a student must have:

1. A minimum grade of C– in all of the following classes or their equivalents:
 - Mathematics 1210/1220 or 111/112/113. (Consult the Mathematics Department for information on satisfying this requirement with a mixture of quarter- and semester-length courses.)
 - Physics 2210/2220 or 221/222/223. (Consult the Physics Department for information on satisfying this requirement with a mixture of quarter- and semester-length courses.)
 - Computer Science 2010/2020 or 201/202/2030. (CS 2030 will be taught only during the fall semester of 1998, and not thereafter. Students who have completed the CS 201/202 sequence under quarters but who plan to apply for full major status during the summer of 1999 or later must take CS 2030 in 1998.)
 - University English writing requirement
2. A grade of CR in CS 1010 or 110 (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 required. See below for details.)

Applicants for the CS major are ranked according to their composite grade point averages in the required classes listed above, and the students with the best composite scores are admitted. The lowest student on the list to be admitted had a score of 3.4 in 1994, 3.3 in 1995, 3.3 in 1996, and 3.35 in 1997. Keep this in mind when estimating your chances for admission.

All classes used in the calculation must be taken for letter grades. *Credit/no-credit grades are not acceptable*, except in CS 110 and CS 1010. Furthermore, each class may be repeated only once. If a class is repeated, only the second grade received is used. If a student registers for a class and later withdraws, resulting in a grade of W, or if a student receives a grade of I or V, that is considered to be one of the two allowable times to register for it.

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 departmental Academic Counselor for details.

1.2 Undergraduate Advising

Each student in the Computer Science major is assigned a faculty advisor. The department also has an Academic Counselor (Mary Rawlinson, 3190 MEB, 581-8224, mrawlins@cs.utah.edu) who is available to answer questions regarding registration for CS classes, transfer of credits, degree requirements, recent departmental 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 B.S. Degree

The Computer Science degree can be completed in four full-time years of study if the student is capable of completing the year-long calculus, physics, and computer science sequences, along with English writing, 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. (If you are classified by the University as a semester transition student, you can elect to satisfy the old Liberal Education requirements instead.)
 - (a) The University writing requirement is required for Computer Science pre-majors.
 - (b) The quantitative reasoning requirement is satisfied by Math 1210/1220 or 111/112/113, 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 or 274, History 1700 or 170, or Political Science 1100 or 110.
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 or 301, which is required for Computer Science majors.
 - (b) The quantitatively intensive course requirement is satisfied by CS 3100 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 300/3000 or above. Two-year college credits 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. (For the purposes of this requirement, one semester hour is equivalent to 1.5 quarter hours.)
3. **Mathematics and Science:** One year of calculus (Mathematics 1210/1220 or 111/112/113) and one year of physics for scientists and engineers (Physics 2210/2220 or 221/222/223) are required. (Contact the relevant department for information on satisfying this requirement with a mixture of quarter- and semester-length courses.)

Students must take three additional classes chosen from among Chemistry 1210 or 121, Biology 1000 or 101, or any math, science, or engineering class that has a full year of calculus as a prerequisite. Students should take the prerequisites of computer science electives into consideration when planning how to satisfy this requirement. (If any of the classes taken are of quarter length, an additional class is required.) At least two of the classes taken to fulfill this requirement must be math classes.
4. **Computer Science:** A minimum of 16 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 3 summarizes the prerequisites for computer science courses.

Example Computer Science Degree Program

| | <i>Fall</i> | | <i>Spring</i> | |
|------------------|---------------------------|---------------|---------------------------|-------------|
| <i>Freshman</i> | Math 1210 [†] | (4) | Math 1220 [†] | (4) |
| | Physics 2210 [†] | (4) | Physics 2220 [†] | (4) |
| | CS 2010 [†] | (4) | CS 2020 [†] | (4) |
| | CS 1010 [†] | (0.5) | Gen Ed | (3) |
| | Writing [†] | (3) | | |
| | | <u>(15.5)</u> | | <u>(15)</u> |
| <i>Sophomore</i> | CS 3500 | (4) | CS 3100 | (3) |
| | CS 3810 | (4) | CS 3510 | (3) |
| | Math elective | (4) | Science elective | (4) |
| | Gen Ed | (3) | Writing 3400 | (3) |
| | | | Gen Ed | (3) |
| | | <u>(15)</u> | | <u>(16)</u> |
| <i>Junior</i> | CS restricted elective | (3) | CS restricted elective | (3) |
| | CS restricted elective | (3) | CS restricted elective | (3) |
| | Math elective | (4) | CS elective | (3) |
| | Gen Ed | (3) | Gen Ed | (3) |
| | Free elective | (3) | Free elective | (3) |
| | | <u>(16)</u> | | <u>(15)</u> |
| <i>Senior</i> | CS elective | (3) | CS capstone | (3) |
| | CS elective | (3) | CS elective | (3) |
| | Gen Ed | (3) | Gen Ed | (3) |
| | Free elective | (3) | Free elective | (3) |
| | Free elective | (3) | Free elective | (3) |
| | | <u>(15)</u> | | <u>(15)</u> |

This table gives an eight-semester example program leading to a B.S. in Computer Science. It is meant only as a guide, since the scheduling of electives and General Education classes depends upon which ones are selected. This schedule assumes adequate high school preparation in college algebra and trigonometry. Note that Math 1210, Physics 2210, and Computer Science 2010 must all be taken during the fall semester in order to complete the required pre-major classes during the first year. Students whose background is not adequate will need to take one or more of Math 1050 or Math 1060. ([†]Class required of pre-majors.)

Figure 1.1: Example Computer Science Degree Program

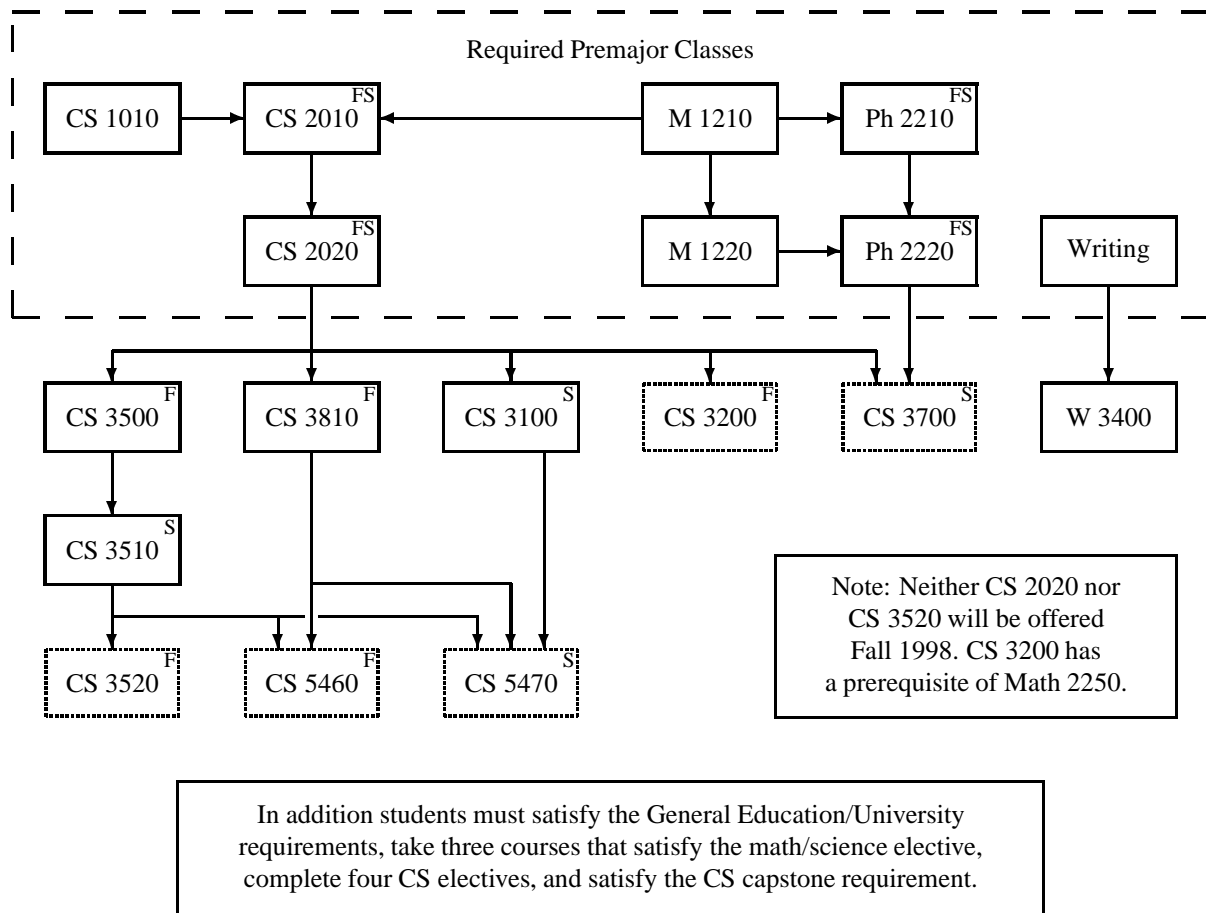
(a) Required. The following classes must be taken:

| | |
|------------------------------|----------------------------------|
| CS 1010 or 110 | Introduction to Unix |
| CS 2010/2020 or 201/202/2030 | Introduction to Computer Science |
| CS 3100 | Models of Computation |
| CS 3500 | Software Practice |
| CS 3510 | Algorithms and Data Structures |
| CS 3810 | Computer Architecture |

(b) Restricted elective. Four of the following five classes must be taken:

| | |
|---------|------------------------|
| CS 3200 | Scientific Computation |
| CS 3520 | Programming Languages |
| CS 3700 | Digital Design |
| CS 5460 | Operating Systems |
| CS 5470 | Compilers |

Computer Science Prerequisites



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. Four of the five courses contained in dashed boxes must be taken. Where not otherwise indicated, courses are offered during both semesters as well as the summer. This graph assumes adequate high school preparation permitting freshman year enrollment in Math 1210/1220, Physics 2210/2220, and Computer Science 2010/2020.

Figure 1.2: Computer Science Prerequisites

- (c) Capstone experience. One of the following two classes must be taken:
- CS 4500 Software Engineering Lab
 - CS 4970 Bachelor's Thesis
- (d) Electives. Four additional Computer Science classes at the 3000 level or higher totaling at least 12 semester hours must be taken.
- (e) Duplication of credit. No single class may be counted toward more than one of the requirements listed above.
5. **Continuing Performance:** 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. Each course taken to satisfy the departmental requirements listed above must be passed with a grade of C– or better. A student may repeat a course at the 3000 level or above only one time. All Computer Science classes (other than CS 1010 or 110) taken to satisfy the requirements for a Computer Science degree must be taken for a letter grade; they may not be taken CR/NC.

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 department rolls. Reinstatement requires a petition to the Computer Science Undergraduate Committee. Reinstated students proceed under the latest graduation requirements.

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 department 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 at the end of the required period of time.

In order to be reinstated from inactive status, 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 in the department, a student may request an extension of a specific duration and submit a revised schedule of completion.

1.4 Computer Science as a Minor

The Department of Computer Science offers a minor for students who desire to gain sufficient background to use and program computers in another field. In addition, a Teaching Minor is offered in conjunction with the College of Education for students who desire to gain sufficient background to teach computer science at the secondary school level.

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. Priority is given to students wishing to pursue a Teaching Minor.

The admission process is similar to that for majors and is carried out at the same time. Students compete for admission based on grades in Math 1210 or 111, English writing, and Computer Science 2010/2020 or 201/202/2030. In order to be accepted, the GPA in these classes must exceed that of students admitted into full-major status that year.

The minor consists of a minimum of 18.5 semester hours of computer science classes. (For the purposes of this requirement, a quarter hour counts as two-thirds of a semester hour.)

- Required. The following classes must be taken.

| | |
|------------------------------|----------------------------------|
| CS 1010 or 110 | Introduction to Unix |
| CS 2010/2020 or 201/202/2030 | Introduction to Computer Science |
| CS 3500 | Software Practice |

- Elective. Students must take at least two additional Computer Science classes at the 3000 level or above.

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 without permission from the Computer Science Undergraduate Committee. 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.

Students who wish to pursue a Teaching Minor should contact an advisor in the College of Education.

1.5 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.

1.6 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 department chairman 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.

1.7 Undergraduate Financial Assistance

The Department of Computer Science has several financial assistance awards available. Applications may be picked up from the Academic Counselor in the Department of Computer Science office. These scholarships are awarded based upon academic performance, rather than financial need. The following scholarships are available:

Special Departmental Scholarships. This award is available to entering Freshmen or transfer students who plan to major in Computer Science or Computer Engineering and who are residents of the state of Utah. The award is for resident tuition for two semesters, renewable up to eight semesters, depending on GPA, for undergraduate work. There are two awards available each year. To be eligible, students must take at least 12 credit hours per semester. Applications for these scholarships must be submitted to the University Office of Financial Aid by February 1.

Continuing Student Departmental Scholarships. This award is available to sophomore, junior or senior students majoring in Computer Science and who are residents of the state of Utah. The award is for resident tuition for two semesters. There are approximately three awards available each year. To be eligible, students must take at least 12 credit hours per semester. Applications for these scholarships must be submitted to the CS departmental office by March 15.

College of Engineering Scholarships. About 10 awards are available each year to Computer Science majors. These are unrestricted cash awards of \$500 to \$1,000. To be eligible, students must take at least 8 credit hours per semester. Use the same application as for the Continuing Student Scholarship.

Sperry Rand Scholarship. This scholarship is awarded on the basis of excellence to one Junior or Senior student majoring in Computer Science or in Computer Engineering. The award is an unrestricted cash prize of approximately \$500. 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.

1.8 Employment Opportunities

The University Placement Center 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, and Unisys. Many of our majors take advantage of this valuable opportunity.

The department 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 quarterly 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 Placement Center in order to be included on a list supplied to employers. Students not planning to work towards an advanced degree should register with the Placement Center at the beginning of their senior year, since most companies begin interviewing in the autumn quarter.

1.9 Student Participation in Departmental 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 department and coordinate the following:

1. Course and faculty teaching evaluations.
2. Representation (one student) at faculty meetings.
3. Newsletters 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 departmental issues affecting students, such as scheduling, curriculum changes, and graduation requirements.

Anyone interested in joining either of these organizations should contact UGSAC at ugsac@cs.utah.edu. Participation, suggestions, and criticisms are solicited.

2

Departmental Facilities

The Departmental Computing Facility is configured to support both instructional and research computing. The instructional facility, or General Computing Facility (GCF), includes more than 130 Unix workstations from Hewlett-Packard, Silicon Graphics, Digital Equipment, and Sun; 30 Hewlett-Packard Pentium/NT machines; and 30 X terminals. These machines are supported by ten file/application servers that provide a total of 200 GByte of disk storage via both NFS file systems and 40 GB of application replicated across five AFS servers. The instructional machines are interconnected with these multi-homed NFS/AFS application/file servers via ethernets, fast ethernet, and FDDI rings. Students in the Department also have access to the College of Engineering's workstation laboratory, which consists of five file/application servers and over 100 Unix workstations from Sun and Hewlett-Packard.

The Research Computing Facility (RCF) includes more than 330 Unix workstations from DEC, HP, Sun, IBM, and SGI; and Pentium NT machines from various vendors. These machines are supported by 20 file/application servers that provide a total of 500 GByte of NFS/AFS disk storage and are supported by the same AFS applications server as in the instructional environment. These common AFS application servers allow the research and instructional computing environment to be easily traversed by both researcher and student. The research computing facility is interconnected with ethernets, fast ethernets, and FDDI rings.

The individual research laboratories contain a wide array of specialized equipment, including

- an SGI Origin 2000 Reality Monster (60 processors, 8 IR heads);
- an SGI Power Challenge (12 processors);
- an SGI Power Onyx (4 processors, 2 RE2 heads);
- a variety of specialized equipment for image analysis, including a GRF-2 light stripper and a DIGI-BOT II laser range scanner;
- two FDDI/Fast Ethernet clusters comprised of IBM RS 6000s, HP 700s, and SGI Indigo2s;
- a collection of graphics workstations from each workstation vendor;
- a multi-source video editing environment;
- robot arms;

- specialized D/A and A/D equipment for realtime signal processing.

The College of Engineering operates a research-scale integrated circuit (IC) fabrication facility that is used extensively by Computer Science. 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 graduate students. The two versions of the class will meet together, but extra work will be expected of graduate students.

Current class schedules and registration information¹ are available on line.

1000 Engineering Computing (3, FS) Coreq: CS 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. (This course is half of a semester in length, and is offered twice during the fall and spring semesters and once during the summer.)

1020 Introduction to Programming (3, U)

An introduction to essential programming concepts using C++. This course is more slowly paced and has much less depth than either CS 1000 (an introductory programming course designed for science and engineering majors) or CS 2010 (an introductory programming course designed for computer science and electrical engineering students).

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

The first course required for students intending to major in computer science. Introduction to the engineering and mathematical skills required to effectively program computers, and to the range of issues confronted by computer

¹<http://www.acs.utah.edu/student/index.htm>

scientists. Roles of procedural and data abstraction in decomposing programs into manageable pieces. Selected topics from discrete math that underlie computer science. Extensive programming exercises that involve the application of elementary software engineering techniques.

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

The second course required for students intending to major in computer science. 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. Selected topics from discrete math that underlie computer science. Extensive programming exercises that require the application of elementary techniques from software engineering. (Not offered Fall 1998.)

2030 Introduction to Computer Science III (2, F) Prereq: CS 202.

Material from the new CS 2010–CS 2020 sequence that could not be covered in the old CS 201–CS 202 sequence. This will include some elementary concepts from discrete math, some of the more advanced data structures and algorithms, and algorithm analysis. (This course will be offered once during the fall semester of 1998, and it will not be taught thereafter. It must be taken by all new Computer Science and Computer Engineering majors at that time.)

3100 Models of Computation (3, S) Prereq: CS 2020.

Introduction to the mathematical underpinnings of computer science. Methods for describing and reasoning about hardware and software, including predicate logic, recursion, induction, and combinatorics. Models of sequential computation, including finite-state automata, push-down automata, and Turing machines. Models of concurrent computation, including Petri nets and communicating sequential processes.

3200 Scientific Computation (3, F) Prereq: CS 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: CS 2020.

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 Algorithms and Data Structures (3, S) Prereq: CS 3500.

Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CS 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: CS 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. (Not offered Fall 1998.)

3700 Fundamentals of Digital System Design (4, S) Prereq: CS 2010, PHYCS 2220. Crosslisted with EE 3700.

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) Prereq: CS/EE 3700, CS/EE 3810. Crosslisted with EE 3710.

Student groups design, build, and test a programmable device such as a computer or calculator. (Not offered Fall 1998.)

3720 Analog & Digital Interfacing with Microprocessors & Microcontrollers (4, S) Prereq: CS/EE 3700. Crosslisted with EE 3720.

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. (Offered Fall 1998, Spring 2000, and every spring semester thereafter. Laboratory included.)

3810 Computer Architecture (4, F) Prereq: CS 2020. Crosslisted with EE 3810.

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. Includes a two-hour laboratory scheduled during the first week of class.

4500 Software Engineering Laboratory (3, S) Prereq: CS 3500 or CS 4510, senior standing in CS.

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

4510 Software Engineering (3, F) Prereq: Senior standing in CS.

Fundamentals of software engineering, including requirements, specifications, design, implementation, testing, and maintenance. Emphasis includes: software process, software engineering techniques (code and documentation standards, source code management, object-oriented analysis and design), and team development practice. (This course will be offered during Fall 1998 and Fall 1999 and will then be discontinued. It is intended for students who completed the CS 354-355-356 sequence under quarters; CS 3500 and CS 4510 may not both be taken for credit.)

4710 Computer Engineering Senior Project (3, F) Prereq: CS/EE 3720. Crosslisted with EE 4710.

Students design microcomputer system that includes RAM, EPROM, and I/O devices. Capstone project for computer-engineering majors. Formal written reports, one or more oral presentations. (Offered Spring 1999, Fall 2000, and every fall semester thereafter.)

4950 Independent Study (Arr.)

4970 Bachelor's Thesis (3) Prereq: Permission of instructor, senior standing in CS.

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

5010 Software Practice (4, F) Prereq: CS 2020 and permission of instructor.

This course is for graduate students from departments other than Computer Science. 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 Algorithms and Data Structures (3, S) Prereq: CS 5010 and permission of instructor.

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

5100/6100 Foundations of Computer Science (3, F) Prereq: CS 3100, CS 3510.

Advanced examination of fundamental ideas behind algorithms, complexity analysis, mathematical logics, elementary computability, and concurrency formalisms.

5210/62120 Advanced Scientific Computing I (3) Prereq: CS 3200, CS 3510, MATH 3160.

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. (Offered every third semester, beginning in Fall 1998.)

5300/6300 Artificial Intelligence (3, F) Prereq: CS 3510.

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.

5310/6310 Robotics (3, F) Prereq: CS 1000, MATH 2250. Crosslisted with ME 5220/6220.

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.

5320/6320 Computer Vision (3, S) Prereq: CS 3510, MATH 2210, MATH 2270.

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.

5340/6340 Natural Language Processing (3, S) Prereq: CS 3510.

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.

5350/6350 Machine Learning (3, F) Prereq: CS 3510.

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. (Offered alternate years, beginning Fall 1999.)

5460/6460 Operating Systems (3, F) Prereq: CS 3510, CS/EE 3810.

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.

5470/6470 Compiler Principles and Techniques (3, S) Prereq: CS 3510, CS/EE 3810, CS 3100.

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.

5480/6480 Data Communications and Networks (3, F) Prereq: CS 3510, CS/EE 3810.

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, nameservers, gateways), network management, distributed applications, network security, and electronic commerce. Principles are put into practice via a number of programming projects.

5520/6520 Programming Languages and Semantics (3, S) Prereq: CS 3520.

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.

5530/6530 Database Systems (3, F) Prereq: CS 3510.

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.

5540/6540 Human/Computer Interaction (3, F) Prereq: CS 3520.

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

5600/6600 Computer Graphics I (3, F) Prereq: CS 3510, MATH 2250.

Basic display techniques, display devices, vector generation, display processors. Homogeneous coordinates, transformations, and clipping in 2-D. Graphics systems, interactive graphics. Introduction to raster graphics. Some elements of photography as related to computer graphics.

5610/6610 Computer Graphics II (3, S) Prereq: CS 5600/6600.

Representations of 3-D objects, polygons, 3-D visualization techniques, hidden-line and hidden- surface removal, polygon clipping, continuous-tone pictures, color displays, lighting models, the aliasing problem. Some fundamentals of photographing computer-generated gray-scale images.

5630/6630 Scientific Visualization (3) Prereq: CS 3510; CS 3200 or CS 5210 or MATH 5600.

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. (Offered every third semester, beginning in Fall 1999.)

5710/6710 Advanced Integrated Circuit Design I (3, F) Prereq: CS/EE 3700. Crosslisted with EE 5710/6710.

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.

5720/6720 Advanced Integrated Circuit Design II (3, S) Prereq: CS/EE 5710/6710, EE 2100. Crosslisted with EE 5720/6720.

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.

5740/6740 Computer-Aided Design of Digital Circuits (3, F) Prereq: CS/EE 3700, CS 3510. Crosslisted with EE 5740/6740.

Introduction to theory 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. (Offered alternate years, beginning Fall 1998.)

5750/6750 Synthesis and Verification of Asynchronous VLSI Systems (3, S) Prereq: CS/EE 3700, CS 3510. Crosslisted with EE 5750/6750.

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. (Offered alternate years, beginning Spring 2000.)

5810/6810 Advanced Computer Architecture (3, F) Prereq: CS/EE 3700, CS/EE 3810. Crosslisted with EE 5810/6810.

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, S) Prereq: CS/EE 3710, CS/EE 3810.

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. (Offered alternate years, beginning Spring 2000.)

5940 Seminar (1-3)

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

5950 Independent Study (Arr.)

5960–5964/6960–6964 Special Topics (Arr.)

The following special topics courses are currently scheduled for the 1998–99 academic year. Contact the faculty member in charge for details.

- **CS 5960/6960 Advanced Compilers** (3, F). Prof. Hsieh.
- **CS 5962 Computers and Law** (2, F). Prof. Hollaar.
- **CS 5963 Advanced Manufacturing** (2,F). Prof. Drake.

6010 Writing Research Proposals (2, S)

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.)

6110 Formal Methods for System Design (3, S) Prereq: CS 5100/6100 and CS 5520/6520.

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. (Offered alternate years, beginning Spring 2000.)

6220 Advanced Scientific Computing II (3) Prereq: CS 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. (Offered every third semester, beginning in Spring 1999.)

6360 Virtual Reality (3, S) Prereq: CS 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. (Offered alternate years, beginning Spring 1999.)

6620 Image Synthesis (3, S) Prereq: CS 5610/6610, CS 6670 MATH 5010.

Using camera and sensor simulation along with physical simulation to generate realistic synthetic images. (Offered alternate years, beginning Spring 1999.)

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

6680 Computer-Aided Geometric Design II (3, S) Prereq: CS 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. (Offered alternate years, beginning Spring 2000.)

6820 Parallel Computer Architecture (3, S) Prereq: CS/EE 5810/6810. Crosslisted with EE 6820.

Architecture, design, and analysis of parallel computer systems: vector processing, data vs. control concurrency, shared memory, message passing, communication fabrics, case studies of current high performance parallel systems. (Offered alternate years, beginning Spring 2000.)

6930–6944 Seminar (1-3)

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

6950 Independent Study (Arr.)

6970 Masters Thesis Research (Arr.)

6980 Faculty Consultation Masters (Arr.)

7120 Information-Based Complexity (3, S) Prereq: CS 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. (Offered alternate years, beginning Spring 1999.)

7240 Sinc Methods (3) Prereq: CS 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. (Not offered 1998–99.)

7310 Advanced Robotics (3, S) Prereq: CS/ME 5310/6310 5220/6220. Crosslisted with ME 7230.

This course covers the kinematics, dynamics, and control of robotic manipulators. Projects controlling robots will be an integral part of the course. (Offered alternate years, beginning Spring 2000.)

7460 Advanced Operating Systems (3) Prereq: CS 5460/6460, CS 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. (Offered alternate years, beginning Spring 2000.)

7970 PhD Dissertation Research (Arr.)

7980 Faculty Consultation PhD (Arr.)

7990 Continuing Registration: Ph.D. (Arr.)

4

CS Faculty And Their Research Interests

Erik Brunvand

Associate Professor of Computer Science
Ph.D., Carnegie Mellon University, 1991

Professor Brunvand¹ joined the Department of Computer Science 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

Assistant Professor of Computer Science
Ph.D., Rice University, 1992

Professor Carter² joined the Department of Computer Science in January 1993. His research interests include multiprocessor computer architecture, operating systems, distributed computing, and computer networks. Of particular interest are scalable shared memory architecture designs, both hardware and software. Dr. Carter is co-leading three DARPA-sponsored research projects: the Avalanche scalable multiprocessor architecture design effort, the Adaptable Memory Systems project, and the Flux operating system project. The goal of the Avalanche project is to develop an integrated cache, memory, and communication architecture that significantly reduces the latency of both distributed shared memory and message passing multiprocessor communication. The goal of the Adaptable Memory Systems 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. The goal of the Flux project is to develop an operating system that provides a much higher degree of flexibility than traditional operating systems, and provides a testbed for exploring other systems issues such as global caching and distributed shared memory.

¹<http://www.cs.utah.edu/~elb/>

²<http://www.cs.utah.edu/~retrac/>

Elaine Cohen

Professor of Computer Science
 Adjunct Associate Professor of Mathematics
 Ph.D., Syracuse University, 1974

Professor Cohen³ has held a faculty position in the Department of Computer Science since 1974. Currently she is co-head of the department'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 of Computer Science
 Ph.D., University of Utah, 1972

Professor Davis⁴ joined the Department in 1993. His research interests include convergence parallel processing system architectures, 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 which contains 19 processing elements and has to date exhibited scalable performance for a wide range of business and scientific applications. Current research interests include the development of low-latency communication protocols and network interface hardware (supported by Hewlett Packard), the design of a scalable parallel processor which supports flexible distributed shared memory and low-latency message passing programming models (supported by ARPA), and a novel adaptive memory system (supported by ARPA).

Ganesh C. Gopalakrishnan

Associate Professor of Computer Science
 Ph.D., State University of New York at Stony Brook, 1986

Professor Gopalakrishnan's⁵ research is primarily in two areas: asynchronous circuit design and formal verification. His asynchronous design group is developing synthesis algorithms to generate asynchronous circuits from descriptions in high-level hardware description languages (currently an enhanced subset of Verilog). A tool embodying these algorithms features user-guided partitioning and complex-gate generation. His formal verification group is involved in the verification of the protocols (pertaining to distributed shared memory management as well as message passing) used in an experimental multiprocessor "Avalanche" under construction at Utah. By using verification as a design aid, we hope to not only detect protocol errors at the earliest stages of design but also provide verified models for synthesis into VLSI circuits. He is also involved in the design of a high-speed image compression chip that uses novel clock distribution methods. He is a member of IFIP working-group 10.5.

David H. Hanscom

Clinical Professor of Computer Science
 Ph.D., Case Western Reserve University, 1970

Professor Hanscom's⁶ 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 program 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 Summer Computing Institute. His interests are in the areas of undergraduate education, computer architecture, and data communications.

³<http://www.cs.utah.edu/~cohen/>

⁴<http://www.cs.utah.edu/~ald/>

⁵<http://www.cs.utah.edu/~ganesh/>

⁶<http://www.cs.utah.edu/~hanscom/>

Thomas C. Henderson

Professor of Computer Science
Ph.D., University of Texas, 1979

Professor Henderson's⁷ professional interests include artificial intelligence, computer vision and robotics. Major areas of current research are robot behavior specification, simulation, multisensor integration and sensing strategies, and parallel vision algorithms. 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), West Germany.

Lee A. Hollaar

Professor of Computer Science
Ph.D., University of Illinois at Urbana-Champaign, 1975

Professor Hollaar's⁸ 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 of Computer Science
Ph.D., Massachusetts Institute of Technology, 1978

Professor Hollerbach's⁹ interests include robotics, teleoperation, virtual reality, and human motor control. 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. In virtual reality, the focus is on improving the transparency and sense of immersion through better mechanical interfaces and their control, better visual and auditory displays, and sensorimotor integration. More specifically, haptic interfaces are being employed for virtual manipulation of Alpha1 CAD models and for scientific visualization. The Treadport locomotion interface is being employed for walk-in synthetic environments. Real-time architectures employing the Myrinet for low-latency networking of devices, computations, and graphics are being developed. Methods for perturbation analysis and nonlinear system identification of joint mechanical properties are being developed, for use in clinical and human operator dynamics settings.

Wilson C. Hsieh

Assistant Professor of Computer Science
Ph.D., Massachusetts Institute of Technology, 1995

Professor Hsieh¹⁰ joined the Department of Computer Science in September 1997. His research interests are in compilers, programming languages, and systems. His primary interest lies in how compiler and programming language technology can be used to build fast, flexible systems—both uniprocessor and multiprocessor systems. For the past few years he has been investigating the use of high-level languages and dynamic code generation in building extensible operating systems. Prior to joining the faculty, Professor Hsieh was a postdoctoral research associate at the University of Washington, where he worked on the SPIN extensible operating system.

⁷<http://www.cs.utah.edu/~tch/>

⁸<http://www.cs.utah.edu/~hollaar/>

⁹<http://www.cs.utah.edu/~jmh/>

¹⁰<http://www.cs.utah.edu/~wilson/>

Christopher R. Johnson

Associate Professor of Computer Science
 Research Assistant Professor of Physics
 Adjunct Assistant Professor of Mathematics and Bioengineering
 Ph.D., University of Utah, 1989

Professor Johnson's¹¹ research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods for partial differential equations, automatic mesh generation, numerical analysis, large scale 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. He directs the Center for Scientific Computing and Imaging and is Co-Director of the Computational Engineering and Science Program.

Robert Kessler

Associate Professor of Computer Science
 Ph.D., University of Utah, 1981

Professor Kessler¹² is director of the Center for Software Science. He is interested in research and development of programming languages – mainly compilers and run-time systems for Lisp, C, and C++, operating systems – work on the Mach and BSD systems, and parallelism – parallel programming. A recent project was just completed in conjunction with the Hewlett Packard Research Labs to design and implement the software system for the Mayfly, a new, distributed memory, parallel processor. The software included two new parallel programming languages: Concurrent Scheme and Distributed C++. Current ARPA-sponsored work is in support of multi-lingual programming and persistence in Lisp. Professor Kessler also has interests in object-oriented programming, and expert system technology applied to software problems. He is Co-Editor-in-Chief of the *International Journal on Lisp and Symbolic Computation* and is general chair for the 1994 Lisp and Functional Programming Conference.

Jay Lepreau

Research Assistant Professor of Computer Science
 B.S., University of Utah, 1983

Professor Lepreau¹³ has been leading research in the Department of Computer Science since 1992 as the Assistant Director of the Computer Systems Laboratory. His research interests focus on operating systems, including many other research areas related to building secure, flexible, and high performance systems. These include information security, programming and domain-specific languages, compilers, networks, distributed systems, and software assurance and engineering. He is Principal Investigator of two DARPA-sponsored research grants focusing on developing a secure, flexible, and high-performance operating system, including user-level but strong management of arbitrary resources, such as memory and the cpu. In this effort, much software has been developed by his group, including the Flick IDL compiler, the OSKit, the Fluke operating system, and the OMOS linker and object server.

Gary E. Lindstrom

Professor of Computer Science
 Ph.D., Carnegie-Mellon University, 1971

Professor Lindstrom's¹⁴ research interests include programming languages, databases, and parallel and distributed computing. 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.

¹¹<http://www.cs.utah.edu/~crj/>

¹²<http://www.cs.utah.edu/~kessler/>

¹³<http://www.cs.utah.edu/~lepreau/>

¹⁴<http://www.cs.utah.edu/~gary/>

Richard F. Riesenfeld

Professor of Computer Science
Ph.D., Syracuse University, 1973

Professor Riesenfeld¹⁵ has been involved in research in the areas of computer graphics, animation, computer aided geometric design and CAD/CAM since joining the Computer Science 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

Assistant Professor of Computer Science
Ph.D., University of Massachusetts at Amherst, 1994

Professor Riloff's¹⁶ research interests are in the areas of natural language processing (NLP), information retrieval, and artificial intelligence. Her current research projects include conceptual sentence analysis, information extraction, corpus-based knowledge acquisition, and text categorization and segmentation. The NLP group at Utah is currently building a conceptual natural language processing system called Sundance, which consists of corpus-based components that can be easily adapted for different domains. Professor Riloff also works on corpus-based systems for generating extraction patterns automatically (the AutoSlog and AutoSlog-TS systems), for building semantic lexicons, and for NLP-based text categorization and segmentation.

Peter Shirley

Assistant Professor of Computer Science
Ph.D., University of Illinois, 1990

Professor Shirley¹⁷ joined the Department 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.

Kris Sikorski

Associate Professor of Computer Science
Ph.D., University of Utah, 1982

Professor Sikorski's¹⁸ 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.

¹⁵<http://www.cs.utah.edu/~rfr/>

¹⁶<http://www.cs.utah.edu/~riloff/>

¹⁷<http://www.cs.utah.edu/~shirley/>

¹⁸<http://www.cs.utah.edu/~sikorski/>

Kent F. Smith

Professor of Computer Science
 Professor of Electrical Engineering
 Ph.D., University of Utah, 1982

Professor Smith's¹⁹ 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 of Computer Science
 Ph.D., University of Alberta, 1965

Professor Stenger's²⁰ research interests include the development of new methods of computation and the computer solution of computationally difficult problems from science and engineering, such as inverse problems, crack problems, flow problems and heat problems. He developed the Sinc methods, which provide optimal algorithms in all areas of engineering computation. He is currently writing, jointly with Michael O'Reilly and Tao Zhang, a *Sinc Tool Box* computer-based tutorial package to make these methods accessible to users. One of his students (Kenneth Parker) has recently completed his Ph.D. dissertation *PTOLEMY: A Sinc-Collocation Mapping Sub-System*, which is a computer sub-package of *Maple* that automates the solution of partial differential equations. Several of his students have recently written or are presently writing computer packages for solving a broad range of difficult engineering problem—such as Navier-Stokes equations (Barkey and Vakili, Narasimhan), Maxwell equations (Naghsh-Nilchi) based on his recently discovered Sinc method of approximating indefinite convolutions, which leads to a unified approach for solving elliptic, parabolic, and hyperbolic differential equations. Another student (Ross Schmittlein) is writing a package based on Sinc, constructing conformal maps. Stenger and his former student O'Reilly have been developing methods which make it possible to invert the Helmholtz equation without computing the forward solution. During the next few academic years he expects to extend these inversion methods so that they can be applied to rendering, to visualization, to the determination of paths for robots, and to the inversion of heat and electromagnetic problems for medical and geophysical applications. Seven of his papers have been accepted for publications during this past academic year.

William B. Thompson

Professor of Computer Science
 Ph.D., University of Southern California, 1975

Professor Thompson's²¹ primary research interest is in the area of computer vision. As part of this work, he has been active in the exploration of techniques for analyzing visual motion. Currently, he is exploring problems in sensing for manufacturing and in vision-based navigation, with a particular interest in how higher-level problem solving and lower-level perception can interact. Professor Thompson joined the University of Utah in 1991 after 16 years in the Computer Science Department at the University of Minnesota.

¹⁹<http://www.cs.utah.edu/~k-smith/>

²⁰<http://www.cs.utah.edu/~stenger/>

²¹<http://www.cs.utah.edu/~thompson/>

Joseph L. Zachary

Clinical Associate Professor of Computer Science
Ph.D., Massachusetts Institute of Technology, 1987

Professor Zachary²² joined the Department 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 and computer science courses. Professor Zachary received 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.

²²<http://www.cs.utah.edu/~zachary/>