COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 10	Course Title	Introduction to Computer Science for Science, Mathematics and Engineering I
Units	4	Course Coordinator	Kris Miller
Required/elective	required	URL (if any):	fish.cs.ucr.edu/moodle

Current Catalog Description: Solving problems through structured programming of algorithms on computers, using the C++ object-oriented language. Topics include variables, expressions, input/output (I/O), branches, loops, functions, parameters, arrays, strings, file I/O, and classes. Also covers software design, testing, and debugging.

Textbook: *Big C++*, by Cay Horstmann and Timothy Budd, John Wiley & Sons, Inc. ISBN: 0-471-47063-5

References/Materials

CodeLab: www.turingscraft.com

Course Goals/Objectives:

1. Use variables to store computer program data
2. Form and use mathematical and Boolean expressions of variables
3. Process program input and generate program output
4. Use branches to create programs incorporating decision making
5. Use loops to create programs that repeat certain behaviors
6. Use functions to modularize programs
7. Use arrays to store collections of data
8. Use strings to handle textual data
9. Use classes as a record that keeps related data together
10. Convert a problem description into a set of about 50-100 computer instructions
11. Debug programs written by oneself or by others
12. Understand very basic methods of testing a program
13. Incorporate useful comments into programs

Prerequisites by Courses and Topics: MATH 009A (may be taken concurrently), First Year Calculus. Introduction to the differential calculus of functions of one variable.

Major Topics Covered in the Course

Variables, expressions, input/output (I/O), branches, loops, functions, parameters, vectors, strings, and object oriented design. Also covers software design, testing, and debugging.

Laboratory schedule: number of sessions per week and duration of each session:

Lecture, 3 hours; laboratory, 3 hours

Laboratory projects (specify number of weeks on each)

Linux operating system and basic programming tools - 1 week
variables and expressions - 1 week
input/output and strings - 1 week
programming with objects (graphics) - 1 week
branches - 1 week
loops - 2 weeks
functions and parameters - 2 weeks
vectors - 1 week
software testing and debugging - part of 2 different lab projects in separate weeks

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Data Structures		
Software Design	20%		Prog. Languages	75%	
Comp. Arch.	5%				

Oral and Written Communications:
Every student is required to submit at least __0__ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students are taught their code should follow the style guidelines of the class or employer they are writing for in order to make them more accessible. 20% of all programming assignment scores is dedicated to enforcing this.

Students are taught the importance of writing their own programs without copying from external sources such as the internet or fellow students. Any unauthorized reuse of code for programming assignments results in a failing grade.

Other than on programming assignments, students are encouraged to work together in teams or study groups. Both in lecture and in the labs, students are encouraged and sometimes rewarded with points for working in teams to solve problems and/or write code. A very small amount of time (15 - 30 mintues total) is spent in lecture discussing the need to develop teamwork skills.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Most of the lecture and especially lab time is spent learning the basic programming concepts listed above and putting them into practice using C++. Approximately 10 minutes out of each hour in lecture is spent teaching the theoretical aspects of each of these topics. For example, to motivate the use of functions about 10 to 20 minutes is spent on explaining how functions allow for code reuse and better organized, more manageable programs. Also, the concept of a function as a "black box" is discussed at this time.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Each lab exercise and programming assignment requires the students to apply their newly learned programming concepts in order to solve problems. Most of these problems are well defined and require a specific solution, but some are intentionally underspecified to force the students to analyze it and ask for clarification.

The last programming assignment asks the student to analyze a partially completed battleship program and then finish it by writing the final two functions.

On written exams, students are required to analyze code written by others and specify its result without the benefit of executing it on a computer.

Solution Design

Please describe the design experiences common to all course sections.

Most of the programming assignments are a build up to a simple moonlander simulation. As each new programming concept is learned, we have them apply that to the moonlander simulation. For example, when variables and expressions are taught, we have them design a program that outputs textually the position at each second of an object in freefall for 4 seconds. We give them the formula for freefall and then they have to apply their knowledge of variables and expressions to calculate and output the position at each second. By the end of the quarter, the last moonlander assignment uses all the concepts they have learned including loops, branches, functions, etc.

Assessment methods: CodeLab Homework 5%, In-lab exercises 5%, In-lab tests 20%, Quizzes 10%, Programming Assignments 10%, Lecture Midterm 20%, Lab Final 10%, Lecture final 20%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Teamwork skills and following a particular coding style guideline to make their program more accessible to others are taught and assessed throughout the course.

Relationship of course to program outcomes: The contribution of CS10 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix

Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially

Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Use variables to store computer program data	2	0	0	0	0	0	0	0	0	0	3
Form and use mathematical and Boolean expressions of variables	3	0	0	0	0	0	0	0	0	0	3
Process program input and generate program output	1	0	2	0	0	0	0	0	0	0	3
Use branches to create programs incorporating decision making	1	0	2	0	0	0	0	0	0	0	3
Use loops to create programs that repeat certain behaviors	1	0	2	0	0	0	0	0	0	0	3
Use functions to modularize programs	1	0	2	0	0	1	0	0	0	0	3
Use arrays to store collections of data	1	0	2	0	0	0	0	0	0	0	3
Use strings to handle textual data	1	0	2	0	0	0	0	0	0	0	3
Use classes as a record that keeps related data together	1	0	2	0	0	0	0	0	0	0	3
Convert a problem description into a set of about 50-100 computer instructions	3	0	3	0	0	0	0	0	0	0	3
Debug programs written by oneself or by others	3	3	0	0	0	0	0	0	0	0	3
Understand very basic methods of testing a program	3	2	0	0	0	0	0	0	0	0	3
Incorporate useful comments into programs	0	0	0	0	0	1	0	0	0	0	3

Prepared by, and date of preparation:
Kris Miller - 6/20/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 12	Course Title	Introduction to Computer Science for Science, Mathematics and Engineering II
Units	4	Course Coordinator	Brian Linard
Required/elective	required	URL (if any):	fish.cs.ucr.edu/moodle

Current Catalog Description: Structured and object-oriented programming in C++, emphasizing good programming principles and development of substantial programs. Topics include recursion, pointers, linked lists, abstract data types, and libraries. Also covers software engineering principles.

Textbook: *Big C++*, by Cay Horstmann and Timothy Budd, John Wiley & Sons, Inc. ISBN: 0-471-47063-5

References/Materials

see course website (fish.cs.ucr.edu/moodle) CodeLab: www.turingscraft.com

Course Goals/Objectives:

1. Use recursion to solve certain programming problems elegantly
2. Use pointers to access data
3. Develop pointer-based linked lists
4. Understand the advantages of abstract data types
5. Use and understand the advantages of libraries
6. Convert a problem description into a set of about 100-to-200 computer instructions
7. Debug programs written by oneself or by others, using a debugger tool
8. Test programs using basic methods
9. Develop basic proficiency of working in a Unix environment

Prerequisites by Courses and Topics: MATH 009A: Introduction to the differential calculus of functions of one variable. May be taken concurrently.

Major Topics Covered in the Course

- classes – interfaces & implementation;
- multiple file compilation & the make utility;
- testing & debugging;
- introduction to OO design and the concept of software engineering;
- pointers & dynamic memory allocation;
- inheritance & polymorphism;
- virtual / pure virtual functions & dynamic binding;
- recursion;
- basic sorting & searching;
- linked lists;
- introduction to the STL.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours

Laboratory projects (specify number of weeks on each)

A lab session (or a half-session) is devoted to each of the topics listed above.

One lab session is a team project: 2 teams compete to build a graphical tic-tac-toe game; each team comprises a project manager, a build master, and teams to implement classes according to provided interfaces.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10		Data Structures	20	
Software Design	25		Prog. Languages	40	
Comp. Arch.	5				

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make __0___ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

No structured treatment of these issues: a total of perhaps 2 hours over the quarter spent in informal discussions, mostly related to the responsibilty of programmers to thoroughly test their code and be willing to vouch for it; and to make their code accessible and readable to their colleagues and successors.
Strict enforcement of plagiarism policies reinforces the value of personal integrity.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Every one of the topics listed above is introduced within a context providing justification and motivation. Some specific "theoretical" treatments are:
the process of abstraction and modelling – 1 hr.
the nature of Object Oriented design – 1 hr.
the role of data structures – 1 hr.
"generic programming" and the STL – 1 hr.

Problem Analysis

Please describe the analysis experiences common to all course sections.

The analysis & design of most assignments is provided for the students in the form of a framework consisting of the interfaces for all required classes, so their main task is the implementation of these interfaces.
However, one major assignment and one lab exercise are intentionally under-specified so that students must analyze the verbal problem descriptions for themselves and seek clarification where needed. These assignments are such that several alternative approaches have to be weighed and the most suitable implemented.

Solution Design

Please describe the design experiences common to all course sections.

See above

Assessment methods:
- 5% for 9 labs (attendance);
- 5% for 8 at-home programming assignments;
- 10% for 8 proctored lab tests on these assignments;
- 10% for 8 quizzes;
- 5% for 8 on-line homeworks;
- 20% midterm;
- 20% proctored programming exercise (3 hr. lab final);
- 25% final exam.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The following topics are situated in the broader context of engineering practice:

- problem analysis
- testing & debugging
- introduction to OO Design & software engineering
- "honoring the contract" specified by a class interface
- the team exercise in lab (see above)

Also, I invite the ACM & IEEE student chapters to address the class, and strongly encourage membership in these organizations.

Relationship of course to program outcomes: The contribution of CS12 to program outcomes (a)-(k) or (1) – (9) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Use recursion to solve certain programming problems elegantly	3	0	0	0	0	0	0	0	0	0	3
Use pointers to access data	1	0	0	0	0	0	0	0	0	0	3
Develop pointer-based linked lists	1	0	0	0	0	0	0	0	0	0	3
Understand the advantages of abstract data types	2	0	0	0	0	0	0	0	0	0	3
Use and understand the advantages of libraries	1	0	1	0	0	0	0	0	0	0	3
Convert a problem description into a set of about 100-to-200 computer instructions	3	0	3	0	0	0	0	0	0	0	3
Debug programs written by oneself or by others, using a debugger tool	3	2	2	0	0	0	0	0	0	0	3
Test programs using basic methods	3	3	2	0	0	0	0	0	0	0	3
Develop basic proficiency of working in a Unix environment	0	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Brian Linard – 6/19/2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 14	Course Title	Introduction to Data Structures and Algorithms
Units	4	Course Coordinator	Ann Gordon-Ross
Required/elective	required	URL (if any):	www.cs.ucr.edu/cs14

Current Catalog Description: Topics include basic data structures such as arrays, lists, stacks, and queues; dictionaries including binary search trees and hashing; priority queues (heaps); introductory analysis of algorithms; sorting algorithms; and object-oriented programming including abstract data types, inheritance, and polymorphism. Also covers solving complex problems through structured software development.

Textbook: *Data Structures and Algorithms in C++,* Michael T. Goodrich, Roberto Tamassia, David Mount.

References/Materials

See course web page – www.cs.ucr.edu/cs14

Course Goals/Objectives:

1. Design and use arrays, lists, stacks and queues, and know when each is most appropriate
2. Design and use binary search trees
3. Design and use hash tables
4. Design and use heaps
5. Understand basic algorithm analysis
6. Be able to design and use several different sorting algorithms, understanding the differences and trade-offs among them
7. Basic understanding of object-oriented programming, including abstract data types, inheritance and polymorphism
8. Convert a problem description into an algorithm that efficiently solves the problem
9. Convert a problem description into a program 200-400 lines long
10. Debug programs written by oneself or by others, using a debugger tool
11. Make extensive use of software tools, including debuggers, in writing programs
12. Know how to thoroughly test programs

Prerequisites by Courses and Topics: CS 011/MATH 011: Introduction to Discrete Structures. Prepositional and predicate calculi, elementary set theory, functions, relations, proof techniques, elements of number theory, enumeration and discrete probability. CS 012: Structured and object-oriented programming in C++, emphasizing good programming principles and development of substantial programs. Topics include recursion, pointers, linked lists, abstract data types, and libraries. Also covers software engineering principles.

Major Topics Covered in the Course

Algorithm and runtime analysis
Algorithm design
Object Oriented design
Linked Lists
Stacks
Queues
Templates
Advanced issues in inheritance, virtual functions, and polymorphism
Trees
Binary search trees
Advanced sorting
Hashing
Priority Queues/heaps
Balanced trees with emphasis on 2-3 trees and red black trees
Advanced STL

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

Each lab consists of a three hour block

Lab 1 – linked list design
Lab 2 – Testing and debugging
Lab 3 – Queues
Lab 4 – Stacks
Lab 5 – Advanced issues with inheritance, operator overloading and virtual functions
Lab 6 – STL
Lab 7 – Sorting
Lab 8 – Hashing
Lab 9 – Heaps
Lab 10 – Advanced STL and algorithms

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	30		Data Structures	55	
Software Design	5		Prog. Languages	5	
Comp. Arch.	5				

Oral and Written Communications:
Every student is required to submit at least __1___ written reports (not including exams, tests, quizzes, or commented programs) of typically __5___ pages and to make __0___ oral presentations of typically _____ minute's duration. Include only material that is

graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

No formal lecture discusses these issues, however, when ever possible I include discussions on these topics during lecture time. I focus on teaching responsibility for ones actions, life-long self-teaching, and the implications of producing inferior products including thorough testing and debugging of code.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Runtime analysis is covered heavily in this course. I spend 2-3 hours of formal presentation and discussion of runtime and theoretical analysis, After those introductory lectures, I spend at least 3-5 minutes each lecture discussing the theoretical analysis for the algorithms and data structures we examine during lecture.

Problem Analysis

Please describe the analysis experiences common to all course sections.

One at home (individual) programming assignment involves runtime analysis of sorting algorithms. Students are given 5 different programs to sort numbers. The students are not told what sorting algorithms they are given. Students are required to create input files to exercise the sorting programs and analyze the runtimes to determine the Big-oh runtime analysis for each executable. Students are required to submit a full research report outlining methodology and results.

Solution Design

Please describe the design experiences common to all course sections.

Four of the five at home (individual) programming assignments involve not only translating a problem description into functional code, but creating well written and efficient object oriented design. Students are graded heavily on their ability to write good code.

Assessment methods: 60% lecture quizzes, midterm and final; 40% lab attendance, assignments and practicals

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

This course deals with the following topics
- Problem analysis
- Program design
- Testing and debugging
- Responsibility as an engineer
- Life-long self-teaching

Relationship of course to program outcomes: The contribution of CS14 to program outcomes (a)-(k) or (1) – (12) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Design and use arrays, lists, stacks and queues, and know when each is most appropriate	2	0	1	0	0	0	0	0	0	0	3
Design and use binary search trees	2	0	1	0	0	0	0	0	0	0	3
Design and use hash tables	2	0	1	0	0	0	0	0	0	0	3
Design and use heaps	2	0	1	0	0	0	0	0	0	0	3
Understand basic algorithm analysis	3	1	1	0	0	0	0	0	0	0	3
Be able to design and use several different sorting algorithms, understanding the differences and trade-offs among them	2	0	3	0	0	0	0	0	0	0	3
Basic understanding of object-oriented programming, including abstract data types, inheritance and polymorphism	0	0	0	0	0	1	0	0	0	0	3
Convert a problem description into an algorithm that efficiently solves the problem	3	3	2	0	0	0	0	0	0	0	3
Convert a problem description into a program 200-400 lines long	2	0	3	0	0	0	0	0	0	0	3
Debug programs written by oneself or by others, using a debugger tool	1	3	0	0	0	0	0	0	0	0	3
Make extensive use of software tools, including debuggers, in writing programs	3	0	0	0	0	0	0	0	0	0	3
Know how to thoroughly test programs	2	3	0	0	0	1	0	0	0	0	0

Prepared by, and date of preparation:
Ann Gordon-Ross – 6/19/2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 61	Course Title	Machine Organization and Assembly Language Programming
Units	4	Course Coordinator	Brian Linnard
Required/elective	elective	URL (if any):	www.cs.ucr.edu/moodle

Current Catalog Description: An introduction to computer organization. Topics include number representation, combinational and sequential logic, computer instructions, memory organization, addressing modes, interrupt, input/output (I/O), assembly language programming, assemblers, and linkers.

Textbook: *Introduction to Computing Systems*, 2nd edition, by Patt & Patel (McGraw-Hill) ISBN 0-07-246750-9

References/Materials

Course website.

Course Goals/Objectives:

- Represent numbers in different bases, including decimal, hexadecimal, and binary, and perform arithmetic on such numbers
- Understand the basic combinational and sequential digital logic components as they relate to understanding the basic parts of a computer, including registers and arithmetic-logic units.
- Understand how computer instructions work for a simple computer addressing modes
- Understand the von Neumann model of computing
- Understand how computer instructions use memory, including different
- Know how interrupts interact with regular computer execution
- Understand modes of input/output
- Understand the roles of assemblers and linkers
- Understand how some Higher Level Language constructs are built in assembly language
- Write assembly language programs of 100-200 instructions

Prerequisites by Courses and Topics: MATH CS 005 or CS 010: Introduction to Computer Science for Science, Mathematics and Engineering I. Solving problems through structured programming of algorithms on computers, using the C++ object-oriented language. Topics include variables, expressions, input/output (I/O), branches, loops, functions, parameters, arrays, strings, file I/O, and classes. Also covers software design, testing, and debugging; or knowledge of programming or consent of instructor.

Major Topics Covered in the Course

13. Binary representations & manipulation of numbers and other information
14. binary logic & its representations in hardware (transistors to gates)
15. combinational logic circuits (adder, decoder, mux)
16. introduction to sequential logic circuits & Finite State Machines
17. introduction to memory system design
18. the von Neumann model, microprocessors & the ISA of the LC-3 microprocessor
19. the complete data path design of the LC-3
20. microprocessor i/o – polling and interrupts
21. (time permitting) detailed analysis of interrupt processing
22. introduction to compilers via analysis of activation record construction
23. Assembly language programming (this is done almost entirely in lab)

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

The entire lab course is devoted to assembly language programming for the LC-3.

- basic assembly language concepts (3 weeks)
- ASCII i/o and multi-digit integer arithmetic (3 weeks)
- array management (1 weeks)
- subroutines and multiple file linking (2 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5		Data Structures	5	
Software Design	10		Prog. Languages	20	
Comp. Arch.	60				

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make __0___ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

No structured treatment of these issues: a total of perhaps 2 hours over the quarter spent in informal discussions, mostly about the immense variety of roles that microprocessors play in the modern world.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

About 10% of the time spent on each topic (see above) is dedicated to locating it in its theoretical context: representations vs the "ding an sich"; introduction to the idea of a Turing Machine; computers as implementations of the von Neumann model; state spaces in the treatment of FSMs; the process of abstraction in analysing systems – especially the microarchitecture and the ISA as two of several layers of a "computer"; HLLs as an abstraction of ML; communications between dissimilar systems in introduction to i/o;

Problem Analysis

Please describe the analysis experiences common to all course sections.

In the assembly language component of the course, students are forced to re-think the notions of programming gleaned from their introduction to a HLL, and recast problems in terms of what a microprocessor can actually do. Typical of this process is their first experience of having to distinguish between the separate ASCII characters used to input a multi-digit integer, the abstract "actual" number those characters represent, and its binary representation in the microprocessor's registers. Other examples involve analyzing the low-level nature of such processes as multiplication; manipulating arrays and memory addresses; managing multiple components of a program via subroutine calls, traps & interrupts.

Solution Design

Please describe the design experiences common to all course sections.

See above (at this level, problem analysis and solution design are not fully distinguisahble phases).

Assessment methods: 15% assignments; 10% exercises; 30% tests; 20% midterm; 25% final

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

I use especially the topics of Turing machines and the von Neumann model to speak about these men as among the "greats" of our discipline, and to enhance the students' sense of their identity as "computer scientists" in that lineage.
Also, I invite the ACM & IEEE student chapters to address the class, and strongly encourage membership in these organizations.

Relationship of course to program outcomes: The contribution of CS61 to program outcomes (a)-(k) or (1) – (10) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially

Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Represent numbers in different bases, including decimal, hexadecimal, and binary, and perform arithmetic on such numbers	3	0	0	0	0	0	0	0	0	0	3
Understand the basic combinational and sequential digital logic components as they relate to understanding the basic parts of a computer, including registers and arithmetic-logic units	3	0	0	0	0	0	0	0	0	0	3
Understand how computer instructions work for a simple computer addressing modes	2	0	0	0	0	0	0	0	0	0	3
Understand the von Neumann model of computing	3	0	0	0	0	0	0	0	0	0	3
Understand how computer instructions use memory, including different	2	0	0	0	0	0	0	0	0	0	3
Know how interrupts interact with regular computer execution	2	0	0	0	0	0	0	0	0	0	3
Understand modes of input/output	1	0	0	0	0	0	0	0	0	0	3
Understand the roles of assemblers and linkers	1	0	0	0	0	0	0	0	0	0	3
Understand how some Higher Level Language constructs are built in assembly language	2	0	0	0	0	0	0	0	0	0	3
Write assembly language programs of 100-200 instructions	3	0	1	0	1	0	0	0	0	0	3

Prepared by, and date of preparation:
Brian Linard, 6/20/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 111	Course Title	Discrete Structures
Units	4	Course Coordinator	Marek Chrobak
Required/elective	required	URL (if any):	

Current Catalog Description: Study of discrete mathematical structures with emphasis on applications to computer science. Topics include asymptotic notation, generating functions, recurrence equations, elements of graph theory, trees, algebraic structures, and number theory. Cross-listed with MATH 111.

Textbook: K.R. Rosen, Discrete Mathematics and its Applications, McGraw-Hill.

References/Materials

Lecture notes.

Course Goals/Objectives:

- To learn how to use correct mathematical terminology and notation.
- To learn the methods of formal mathematical reasoning and proof techniques, including proofs by contradiction and by induction.
- To learn how to model real-life problems using discrete mathematical structures: sets, sequences, combinations, permutations, graphs, trees, relations, and algebraic structures.
- To master the concept of asymptotic notation and its application to estimating running time of various algorithms.
- To learn fundamentals of number theory and its applications to cryptographic protocols.
- To learn techniques for solving recurrence equations, and their applications to counting and to analyzing the complexity of divide-and-counter algorithms.
- To learn the basic concepts in graph theory, including connectivity, cycles, planarity, coloring.

Prerequisites by Courses and Topics: CS 010: Introduction to Computer Science for Science, Mathematics and Engineering I; CS 011/MATH 011: Introduction to Discrete Structures; MATH 009C or MATH 09HC: First Year Calculus; Introduction to the integral calculus of functions of one variable..

Major Topics Covered in the Course: Asymptotic notation: asymptotic behavior of functions, polynomial, exponential, and logarithmic functions. Number theory: modular arithmetic, Chinese Remaider Theorem, Euler's Theorem, the RSA. Advanced counting: inclusion-exclusion, generating functions, solving recurrence equations. Elements of graph theory: undirected, directed graphs, connectivity, planarity, Euler cycles, spanning trees. Trees. Algebraic structures: monoids, groups, more modular arithmetic, permutation groups, rings, homomorphisms, isomorphisms, quotient structures. Possible other topics: error-correcting codes, matrix games

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; discussion, 1 hour.

Laboratory projects (specify number of weeks on each)

None.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	25%		Data Structures		
Software Design			Prog. Languages		
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least ___0__ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

N/A

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Discrete mathematics, 100%

Problem Analysis
Please describe the analysis experiences common to all course sections.

Students model real-life problems and solve them using discrete mathematical structures. Most of these problems arise in computer applications: for example cryptography, asymptotic analysis of algorithms, and other.

Solution Design

Please describe the design experiences common to all course sections.

The class does not cover engineering design as such. But the class provides background needed for the design of data structures in algorithms that are covered in CS141.

Assessment methods: 15% assignments; 10% exercises; 30% tests; 20% midterm; 25% final

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

N/A

Relationship of course to program outcomes: The contribution of CS111 to program outcomes (a)-(k) or (1) – (7) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
To learn how to use correct mathematical terminology and notation.	3	0	0	0	0	0	0	0	0	0	3
To learn the methods of formal mathematical reasoning and proof techniques, including proofs by contradiction and by induction.	3	0	0	0	0	0	0	0	0	0	3
To learn how to model real-life problems using discrete mathematical structures: sets, sequences, combinations, permutations, graphs, trees, relations, and algebraic structures.	2	0	0	0	0	0	0	0	0	0	3
To master the concept of asymptotic notation and its application to estimating running time of various algorithms.	3	0	0	0	0	0	0	0	0	0	3
To learn fundamentals of number theory and its applications to cryptographic protocols.	2	0	0	0	0	0	0	0	0	0	3
To learn techniques for solving recurrence equations, and their applications to counting and to analyzing the complexity of divide-and-counter algorithms.	2	0	0	0	0	0	0	0	0	0	3

To learn the basic concepts in graph theory, including connectivity, cycles, planarity, coloring.	1	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation: Marek Chrobak, June 15 2006

COURSE DESCRIPTION

Dept., Number	EE/CS 120A	Course Title	Intro. to Embedded Systems
Units	5	Course Coordinator	Frank Vahid / Sheldon Tan
Required/elective	Req	URL (if any):	www.cs.ucr.edu/cs120a

Current Catalog Description

Design of digital systems. Topics include Boolean algebra; combinational and sequential logic design; design and use of arithmetic-logic units, carry-lookahead adders, multiplexors, decoders, comparators, multipliers, flip-flops, registers, and simple memories; state-machine design; and basic register-transfer level design. Laboratories involve use of hardware description languages, synthesis tools, programmable logic, and significant hardware prototyping. Cross-listed with CS/EE 120A.

Textbook

Required: Digital Design, Frank Vahid, J. Wiley and Sons, ISBN 0-471-46784-7

References/Materials

Logic and Computer Design Fundamentals and Xilinx Student Edition 4.2 Package, 3/E , by Mano and Kime, 3rd edition

Course Goals/Objectives

- Able to perform the conversion among different number systems; familiar with basic logic gates – AND, OR & NOT, XOR, XNOR; independently or work in team to build simple logic circuits using basic.
- Understand Boolean algebra and basic properties of Boolean algebra; able to simplify simple Boolean functions by using the basic Boolean properties.
- Able to design simple combinational logics using basics gates. Able to optimize simple logic using Karnaugh maps, understand “don’t care”.
- Familiar with basic sequential logic components: SR Latch, D Flip-Flop and their usage and able to analyze sequential logic circuits.
- Understand finite state machines (FSM) concept and work in team to do sequence circuit design based FSM and state table using D-FFs.
- Familiar with basic combinational and sequential components used in the typical datapath designs: Register, Adders, Shifters, Comparators; Counters, Multiplier, Arithmetic-Logic Units (ALUs), RAM. Able to do simple register-transfer level (RTL) design.
- Able to understand and use one high-level hardware description languages (VHDL or Veriliog) to design combinational or sequential circuits.
- Understand that the design process for today’s billion-transistor digital systems becomes a more programming based process than before and programming skills are important.

Prerequisites by Courses and Topics

CS 61 machine organization

Major Topics Covered in the Course

Topics include Boolean algebra; combinational and sequential logic design; design and use of arithmetic-logic units, carry-lookahead adders, multiplexors, decoders, comparators, multipliers, flip-flops, registers, and simple memories; state-machine design; and basic register-transfer level design. Laboratories involve use of hardware description languages, synthesis tools, programmable logic, and significant hardware prototyping.

Laboratory schedule: number of sessions per week and duration of each session

Two three-hour sessions per week

Laboratory projects (specify number of weeks on each)

Lab 1: Hardware breadboarding (1 week)
Lab 2: Schematic editor and simulation tools (1 week)
Lab 3: Hierarchical design and technology mapping (1 week)
Lab 4: Carry lookahead adder and multiplier (1 week)
Lab 5: Combinational logic design (1 week)
Lab 6: Latches and flip-flops (1 week)
Lab 7: Designing sequential circuits (1 week)
Lab 8: VGA monitor controller design (2 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	15%		Data Structures	10%	
Software Design	20%		Prog. Languages	20%	
Comp. Arch.	35%				

Oral and Written Communications:

Every student is required to submit at least __8___ written reports (not including exams, tests, quizzes, or commented programs) of typically __2___ pages and to make __0___ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Boolean algebra – 6 lecture hours
Finite-state machines – 3 lecture hours

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analysis of size and performance of two-level and multi-level logic circuits

Solution Design

Please describe the design experiences common to all course sections.

Lecture: Capturing desired circuit behavior using Boolean equations, truth tables, finite-state machines, or high-level state machines, and converting such behavior to circuits of gates and registers.
Lab: Design of circuits using schematic capture and using tools to map to FPGAs.

Assessment methods

Lecture: Quizzes, midterm exam, final exam, homeworks, and in-lecture exercises
Lab: Demos, reports, code

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

This course allows students to understand the internal design of computer processors and to practice in design and optimization of logic circuits both theoretically and experimentally. Also, Teaches use of modern tools, and teaches hands-on debugging and troubleshooting.

Relationship of course to program outcomes: The contribution of Engr 180 to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K

Able to perform the conversion among different number systems; familiar with basic logic gates – AND, OR & NOT, XOR, XNOR; independently or work in team to build simple logic circuits using basic.	2	3	3	2	2	1	1	1	0	0	3
Understand Boolean algebra and basic properties of Boolean algebra; able to simplify simple Boolean functions by using the basic Boolean properties.	2	3	3	2	2	1	1	1	0	0	3
Able to design simple combinational logics using basics gates. Able to optimize simple logic using Karnaugh maps, understand "don't care".	2	3	3	2	2	1	1	1	0	0	3
Familiar with basic sequential logic components: SR Latch, D Flip-Flop and their usage and able to analyze sequential logic circuits.	2	3	3	2	2	1	1	1	0	0	3
Understand finite state machines (FSM) concept and work in team to do sequence circuit design based FSM and state table using D-FFs.	2	3	3	2	2	1	1	1	0	0	3
Familiar with basic combinational and sequential components used in the typical datapath designs: Register, Adders, Shifters, Comparators; Counters, Multiplier, Arithmetic-Logic Units (ALUs), RAM. Able to do simple register-transfer level (RTL) design.	2	3	3	2	2	1	1	1	0	0	3
Able to understand and use one high-level hardware description languages (VHDL or Veriliog) to design combinational or sequential circuits.	0	0	3	0	0	0	0	0	0	0	3
Understand that the design process for today's billion-transistor digital systems becomes a more programming based process than before and programming skills are important.	0	0	3	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Tom Payne with help from Harry Hsieh, Vladimir Fonoberov, and Victor Hill, 6/27/06

COURSE DESCRIPTION

Dept., Number	EE/CS 120B	Course Title	Intro. to Embedded Systems
Units	5	Course Coordinator	Frank Vahid / Harry Hsieh
Required/elective	Req	URL (if any):	www.cs.ucr.edu/cs120b

Current Catalog Description:
Introduction to hardware and software design of digital computing systems embedded in electronic devices (such as digital cameras or portable video games). Topics include custom and programmable processor design, standard peripherals, memories, interfacing, and hardware/software tradeoffs. Laboratory involves use of synthesis tools, programmable logic, and microcontrollers and development of working embedded systems. Cross-listed with EE 120B.

Textbook

Required: "Embedded System Design: A Unified Hardware/Software Approach", Vahid and Givargis, Wiley & Sons, 2002 Required: VHDL : A Starter's Guide (2nd Edition)", Sudhakar Yalamanchili, ISBN 0131457357, Prentice Hall, 2004

References/Materials

Recommended: "C and the 8051 -- 3rd edition", Thomas Schultz, ISBN 1-58961-237-X, Pagefree Publishing Inc, 2004. Recommended: The C Programming Language, Second Edition, Kernighan and Ritchie, Prentice Hall, ISBN 0-13-110362-8

Course Goals/Objectives

1: Understand chip technology trends Moore's law the nature of embedded computing the need to balance competing design factors and the growing productivity gap.
2: Calculate estimated time and cost impacts of various design decisions
3: Describe system behavior as a state machine and design a controller digital circuit implementation
4: Describe system behavior as extended state machines and design a custom processor (controller and datapath)implementation
5: Describe system behavior as a sequential algorithm and design a custom processor implementation
6: Understand basic pipelining and hazards
7: Design a basic but working instruction-set processor
8: Understand assembly language and write simple assembly-level programs
9: Understand the function design and use of common peripherals: timers UART PWM A2D D2A converters.
10: Understand communication methods including I/O schemes interrupts direct-memory access and arbitration
11: Draw timing diagrams to represent communication
12: Understand hardware/software trade-off through examples.
13: Write VHDL and program FPGAs write assembly and C code for microcontroller and build embedded systems in a laboratory environment.
14: Make short presentations in class about contemporary topics concerning embedded systems such as security energy novel circuit structures graphic interfaces ubiquitous computing and ethics.

Prerequisites by Courses and Topics:
Lecture, 3 hours; laboratory, 6 hours. Prerequisite(s): CS 120A/EE 120A.

Major Topics Covered in the Course

Custom and programmable processor design, standard peripherals, memories, interfacing, and hardware/software tradeoffs. Laboratory involves use of synthesis tools, programmable logic, and microcontrollers and development of working embedded systems.

Laboratory schedule: number of sessions per week and duration of each session

Two sessions per week, three hours per session.

Laboratory projects (specify number of weeks on each)

Lab 1: Introduction to VHDL and Aldec-HDL (0.5 weeks)
Lab 2: Simple ALU (0.5 weeks)
Lab 3: Introduction to the Digilent D2SB (1 week)
Lab 4: Sequential Circuits (1 week)
Lab 5: Converting an Algorithm to a Circuit (2 weeks)
Lab 6: Introduction to the 8051 Microcontroller (0.5 weeks)
Lab 7: Using the 8051 to Interface to a Keypad (1.5 weeks)
Lab 8: FPGA and 8051 Tone Generation (2 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	15%		Data Structures	10%	
Software Design	20%		Prog. Languages	20%	
Comp. Arch.	20%	15%			

Oral and Written Communications:
Every student is required to submit at least __7___ written reports (not including exams, tests, quizzes, or commented programs) of typically __2___ pages and to make __2___ oral presentations of typically __5___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

First week discusses (in lecture and in textbook) subject of applying embedded systems for improved quality of life and safety, including discussion of systems like medical devices, automobiles, etc.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Finite-state machines – 3 lecture hours
High-level finite state machines – 3 lecture hours

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analysis of size and performance of custom-designed processing circuits, compared with general-purpose processing circuits.

Solution Design

Please describe the design experiences common to all course sections.

Lecture: Design of custom-processing circuits from high-level behavioral descriptions. Design of computing systems consisting of microprocessor, memory, and peripherals. Lab: Design of circuits using VHDL and tools to map to FPGAs. Design of programs using C using C compilation tools to map to microcontrollers.

Assessment methods

Lecture: Quizzes, midterm exam, final exam, homeworks, and in-lecture exercises Lab: Demos, reports, code, and lab practical exam

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Emphasizes tradeoffs among different implementation options, especially tradeoffs among custom processing circuits ("hardware") and programs on general-purpose processors ("software").

Relationship of course to program outcomes: The contribution of CS120b to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand chip technology trends Moore's law the nature of embedded computing the need to balance competing design factors and the growing productivity gap.	3	0	0	0	0	0	0	0	0	2	3
Calculate estimated time and cost impacts of various design decisions	3	0	0	0	0	1	0	0	0	1	3
Describe system behavior as a state machine and design a controller digital circuit implementation	3	0	0	0	0	0	0	0	0	0	3
Describe system behavior as extended state machines and design a custom processor (controller and data path)implementation	3	0	0	0	0	0	0	0	0	0	3
Describe system behavior as a sequential algorithm and design a custom processor implementation	3	0	0	0	0	0	0	0	0	0	3
Understand basic pipelining and hazards	1	0	0	0	0	0	0	0	0	0	3
Design a basic but working instruction-set processor	3	0	0	0	0	0	0	0	0	0	3
Understand assembly language and write simple assembly-level programs.	1	0	0	0	0	0	0	0	0	0	3
Understand the function design and use of common peripherals: timers UART PWM A2D D2A converters.	2	0	0	0	0	0	0	0	0	0	3
Convert a problem description into a set of about 50-100 computer instructions	1	0	0	0	0	0	0	0	0	0	3
Understand communication methods including I/O schemes interrupts direct-memory access and arbitration	1	0	0	0	0	0	0	0	0	0	3
Draw timing diagrams to represent communication	1	0	1	0	1	0	0	0	0	0	3
Understand hardware/software trade-off through examples.	1	0	1	0	1	0	0	0	0	0	3
Write VHDL and program FPGAs write assembly and C code for microcontroller and build embedded systems in a	3	0	0	0	0	0	0	0	0	0	3

laboratoryenvironment.											
Make short presentations in class about contemporary topics concerning embedded systems such as security energy novel circuit structures graphic interfaces ubiquitous computing and ethics.	1	0	0	0	0	2	3	0	0	2	3

Prepared by, and date of preparation:
Frank - 06/15/06

COURSE DESCRIPTION

Dept., Number	CS 122A	Course Title	Intro. to Embedded Systems
Units	5	Course Coordinator	Frank Vahid / Harry Hsieh
Required/elective	Req	URL (if any):	www.cs.ucr.edu/cs122a

Current Catalog Description

Covers software and hardware design of embedded computing systems. Topics include hardware and software code sign, advanced programming paradigms including state machines and concurrent processes, real-time programming and operating systems, basic control systems, and modern chip and design technologies. Laboratories involve use of microcontrollers, embedded microprocessors, programmable logic and advanced simulation, and debug environments.

Textbook

"C and the 8051 -- 3rd edition", Thomas Schultz, ISBN 1-58961-237-X, Pagefree Publishing Inc, 2004. www.CAndThe8051.com
"VHDL : A Starter's Guide (2nd Edition)", Sudhakar Yalamanchili, ISBN 0131457357, Prentice Hall, 2004.

References/Materials

The C Programming Language, Second Edition, Kernighan and Ritchie, Prentice Hall, ISBN 0-13-110362-8.

Course Goals/Objectives

1: Understand competing design metrics and cost and time implications of various design decisions
2: Understand hardware/software trade-offs
3: Be able to use and choose among different behavior models like FSMD sequential programs HCFSM and dataflow languages for describing system behavior.
4: Understand basic concurrent process execution synchronization and communication methods
5: Compute valid and analyze invalid real-time schedules with techniques such as Rate-Monotonic Scheduling and Earliest-Deadline-First Scheduling.
6: Be able to recognize systems that represent Open-Loop and Closed-Loop control systems.
7: Design and use PID controllers.
8: Understand basic two-level and multi-level logic minimization techniques and apply to simple logic equations.
9: Understand reliability issues and how to use redundancies.
10: Write VHDL and program FPGA and FPGA platforms. Write assembly and C code for microcontroller and multimedia processor utilize the peripherals and build embedded systems in a laboratory environment.
11: Make short presentations in class about contemporary topics concerning embedded systems such as ubiquitous computing energy wireless security nano-tech novel circuit structures government and ethics.

Prerequisites by Courses and Topics

CS 012, CS 120B/EE 120B

Major Topics Covered in the Course

Covers software and hardware design of embedded computing systems. Topics include hardware and software code sign, advanced programming paradigms including state machines and concurrent processes, real-time programming and operating systems, basic control systems, and modern chip and design technologies. Laboratories involve use of microcontrollers, embedded microprocessors, programmable logic and advanced simulation, and debug environments.

Laboratory schedule: number of sessions per week and duration of each session

Two sessions per week, three hours per session.

Laboratory projects (specify number of weeks on each)

Lab 1 (1 week): VHDL simulation and synthesis, FPGAs, debug tools and methods
Lab 2 (1 week): Driver for the 4x7-segment LEDs using time multiplexing
Lab 3 (1 week): Test and debug existing code – Broken PS2 keyboard interface
Lab 4 (1 week): Microcontroller programming – Interfacing a microcontroller with an LCD
Lab 5 (1 week): Microcontroller programming, timers and interrupts – Reflex Timer
Project (5 weeks): Course project: DDR-like game involving Playstation controller input, LED/LCD display, synchronization with music, timing-based scoring, etc.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	15%		Data Structures	10%	
Software Design	10%	10%	Prog. Languages	10%	10%
Comp. Arch.	20%	15%			

Oral and Written Communications:

Every student is required to submit at least __7___ written reports (not including exams, tests, quizzes, or commented programs) of typically __3___ pages and to make __4___ oral presentations of typically __7___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

First week discusses (in lecture and in textbook) subject of applying embedded systems for improved quality of life and safety, including discussion of systems like medical devices, automobiles, etc. Specific lecture covers designing safe systems.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Finite-state machines – 1 lecture hour
High-level finite state machines – 2 lecture hours
Concurrent process model – 2 lecture hours
Computation models versus programming languages – 2 lecture hours

Problem Analysis

Please describe the analysis experiences common to all course sections.

Extensive analysis of size and performance of custom-designed processing circuits, compared with general-purpose processing circuits. Analysis of design time to build different types of circuits using different technologies. Extensive tradeoff analysis throughout course.

Solution Design

Please describe the design experiences common to all course sections.

Lecture: Design of custom-processing circuits from high-level behavioral descriptions. Design of computing systems consisting of microprocessor, memory, and peripherals. Design of time-constrained programs using interrupts. Design of FPGA fabrics.
Lab: Design of circuits using VHDL and tools to map to FPGAs. Design of programs using C using C compilation tools to map to microcontrollers. Consideration of tradeoffs among those two implementation styles to create a final working project consisting of both.

Assessment methods

Lecture: Quizzes, midterm exam, final exam, homeworks, and in-lecture exercises
Lab: Demos, reports, code.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Emphasizes tradeoffs among different implementation options, especially tradeoffs among custom processing circuits (“hardware”) and programs on general-purpose processors (“software”).

Relationship of course to program outcomes: The contribution of CS122a to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand competing design metrics and cost and time implications of various design decisions	2	0	1	0	0	0	0	0	0	0	3
Understand hardware/software trade-offs	2	0	1	0	0	0	0	0	0	0	3
Be able to use and choose among different behavior models like FSMD sequential programs HCFSM and dataflow languages for describing system behavior.	2	0	2	0	2	0	0	0	0	0	3
Understand basic concurrent process execution synchronization and communication methods	2	0	0	0	0	0	0	0	0	0	3
Compute valid and analyze invalid real-time schedules with techniques such as Rate-Monotonic Scheduling and Earliest-Deadline-First Scheduling.	3	1	0	0	0	0	0	0	0	0	3
Be able to recognize systems that represent Open-Loop and Closed-Loop control systems.	1	0	0	0	1	0	0	0	0	0	3
Design and use PID controllers.	3	0	2	0	2	0	0	0	0	0	3
Understand basic two-level and multi-level logic minimization techniques and apply to simple logic equations.	2	0	0	0	0	0	0	0	0	0	3
Understand reliability issues and how to use redundancies.	1	0	0	0	0	0	0	0	0	0	3
Write VHDL and program FPGA and FPGA platforms. Write assembly and C code for microcontroller and multimedia processor utilize the peripherals and build embedded systems in a laboratory environment.	3	1	1	0	0	0	0	0	0	0	3
Make short presentations in class about contemporary topics concerning embedded systems such as ubiquitous computing energy wireless security nano-tech novel circuit structures government and ethics.	0	0	0	0	0	2	3	0	0	2	3

Prepared by, and date of preparation:
Harry - 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 122B	Course Title	Advanced Embedded and Real-Time Systems
Units	5	Course Coordinator	Harry Hsieh / Frank Vahid
Required/elective	elective	URL (if any):	www.cs.ucr.edu/~harry/cs122b

Current Catalog Description: Further exploration of state-of-the-art aspects of building embedded computer systems. Topics include real-time programming, synthesis of coprocessor cores, application-specific processors, hardware and software co-simulation and co-design, low-power design, reconfigurable computing, core-based design, and platform-based methodology.

Textbook:

References/Materials

Embedded System Design: A Unified Hardware/Software Approach, Vahid and Givargis, Wiley & Sons, 2002, Accompanying Web page .

Real Time Systems and Programming Languages: Ada 95, Real-Time Java and Real-Time C/POSIX by Alan Burns, Andrew J. Wellings, 3rd edition (April 5, 2001) Addison-Wesley Publishing.

Embedded System Design by Peter Marwedel, Kluwer Academic Publishers, 2003, ISBN 1-4020-7690-8.

Course Goals/Objectives:

- Be able to write small real-time program utilizing semaphores and other synchronization constructs.
- Be able to write system level program utilizing the concept of modules channels and other system level entities.
- Understand scheduling for concurrent processes
- Understand performance/size/power trade-off in the embedded architecture
- Understand and learn to use Application Specific Instruction Processors

Prerequisites by Courses and Topics: CS 012: Introduction to Computer Science for Science, Mathematics and Engineering II, CS 122A: Intermediate to Embedded and Real-Time Systems. Covers software and hardware design of embedded computing systems. Topics include hardware and software co-design, advanced programming paradigms including state machines and concurrent processes, real-time programming and operating systems, basic control systems, and modern chip and design technologies. Laboratories involve use of microcontrollers, embedded microprocessors, programmable logic and advanced simulation, and debug environments.

Major Topics Covered in the Course

Lecture topic includes method and technology of embedded system design, how to deal with hardware/software/firmware, embedded system specification, real-time scheduling and programming, synchronization and atomic actions, system-level transformation, low-power design, and software testing and reliability. Lab topics include synchronization and embedded OS, programming with application specific instruction processor (both standard and customizable), using system description languages, and using platform FPGA.	

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 6 hours

Laboratory projects (specify number of weeks on each)

24. Tornado: (1.5 weeks) programming with embedded OS for synchronization of concurrent processes.
25. Trimedia: (1.5 weeks) programming ASIP for DVD application.
26. SystemC: (1.5 weeks) programming with system design language.
27. Microblaze: (1.5 weeks) programming platform FPGA.
28. Tensilica: (1.5. weeks) programming flexible ISA for performance/power trade-off.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		20%	Data Structures		10%
Software Design	5%	15%	Prog. Languages	10%	20%
Comp. Arch.		20%			

Oral and Written Communications:
Every student is required to submit at least ___9__ written reports (not including exams, tests, quizzes, or commented programs) of typically __4___ pages and to make ___1__ oral presentations of typically ____15_ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

First week discusses (in lecture) subject of applying embedded systems for improved quality of life and safety, including discussion of systems like medical devices, automobiles, robots, etc. Student presentations include application of embedded system to socially related projects (e.g. earthquake alarm, alternative energy sources, medical robots).

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Embedded system application and method of design – 1 lecture hour
Formal Specification (StateChart/SDL/PetriNet/MSC/UML/PN/SDL) – 5 lecture hours
Real-time Scheduling and synchronization of concurrent processes – 5 lecture hours
High level optimization – 2 lecture hours
Low Power Techniques – 2 lecture hours

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analysis of area/performance/power/design-time/reliability of advance embedded system design which may consists of both hardware/software solution on the same implementation. Extensive trade-off analysis throughout.

Solution Design

Please describe the design experiences common to all course sections.

Lecture: Design of advance embedded systems (consists of hardware/software/firmware) from high-level behavioral descriptions. Design of concurrent systems, design of low-power systems and high reliable system.
Lab: Design of concurrent systems using Embedded OS, ASIP, high level behavioral description, platform-FPGA, and customizable ASIP.

Assessment methods:

Lecture: Midterm exam, final exam, homeworks.

Lab: Demo, reports, code, and exam.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Emphasizes tradeoffs among different implementation options, especially tradeoffs among use of high level description languages, different scheduling algorithms, different techniques for low power and reliability, different customizable instructions set architecture. The metrics for trade-offs are performance/power/area/design_time/reliability.

Relationship of course to program outcomes: The contribution of CS122B to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Be able to write small real-time program utilizing semaphores and other synchronization constructs.	2	0	1	0	0	0	0	0	0	2	3
Be able to write system level program utilizing the concept of modules channels and other system level entities.	2	0	1	0	0	0	0	0	0	0	3
Understand scheduling for concurrent processes	2	0	2	0	2	0	0	0	0	0	3
Understand performance/size/power trade-off in the embedded architecture	2	0	0	0	0	0	0	0	0	0	3
Understand and learn to use Application Specific Instruction Processors	3	1	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation: Harry Hsieh, June 15 2006

COURSE DESCRIPTION

Dept., Number	cs130	Course Title	
Units		Course Coordinator	v zordan
Required/elective		URL (if any):	www.cs.ucr.edu/~vbz/cs130.html

Current Catalog Description

A study of the fundamentals of computer graphics necessary to design and build graphics applications. Examines raster graphics algorithms including scan-converting graphics primitives, anti-aliasing, and clipping. Also covers geometric transformations, viewing, solid modeling techniques, hidden-surface removal algorithms, color models, illumination, and shading. Individual and group projects are assigned.

Textbook

Computer Graphics with OpenGL by Hearn and Baker

References/Materials

n/a

Course Goals/Objectives

1: To understand the software and hardware processes involved in the creation of synthetic imagery along with difficulties and limitations unique to such discrete images.
2: To obtain a working knowledge of fundamental algorithms which allow description and display of graphics primitives including lines, curves, polygons, and surfaces.
3: To obtain a working knowledge of 3D transformations, including simple affine, perspective, and composite transformations.
4: To become familiar with simple synthetic illumination models and fundamental techniques for rendering synthetic images from 3D scenes as well as to be exposed to aspects related to movement within these scenes.

Prerequisites by Courses and Topics

Prerequisite(s): CS 141, MATH 113 (MATH 113 may be taken concurrently); or consent of instructor.

Major Topics Covered in the Course

geometry representations and transformations; rendering and light models; rasterization and anti-aliasing techniques; and basic animation.

Laboratory schedule: number of sessions per week and duration of each session

1 x 3 hour session per week.

Laboratory projects (specify number of weeks on each)

week 1 intro to opengl - draw lines
week 2 intro to maya - modeling/UI
week 3 solid constructive geometry (CSG) in maya
week 4 maya scripting language
week 5 prep/open lab for project 1
week 6 opengl cameras/primitives
week 7 maya rendering and shading
week 8 prep/open lab for project 2
week 9 maya procedural animation
week 10 make-up

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	25	15	Data Structures	5	
Software Design	10	15	Prog. Languages	10	10
Comp. Arch.	10				

Oral and Written Communications:

Every student is required to submit at least ___0__ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

n/a

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

geometry manipulation - 2.5 lectures
matrix transformation - 2 lectures
applied linear algebra - 1.5 lecture
physics of light transfer - 2 lectures

Problem Analysis

Please describe the analysis experiences common to all course sections.

problem solving in class, in projects, in exams related to geometry, physics, motion control

Solution Design

Please describe the design experiences common to all course sections.

two - three programming intensive projects requiring independent design and implementation

Assessment methods

Programs (2 x 15% = 30%)
Laboratory (15%)
Quizzes (2 x 15% = 30%)
Final exam (25%)

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

must think abstractly about various scientific and engineering principles related to graphical primitives and efficient algorithms and then structure these thought into concrete implementation

Relationship of course to program outcomes: The contribution of CS130 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
To understand the software and hardware processes involved in the creation of synthetic imagery along with difficulties and limitations unique to such discrete images.	3	0	1	0	1	0	0	0	0	0	3
To obtain a working knowledge of fundamental algorithms which allow description and display of graphics primitives including lines curves polygons and surfaces.	3	0	0	0	0	0	0	0	0	0	3
To obtain a working knowledge of 3D transformations including simple affine perspective and composite transformations.	3	0	0	0	0	0	0	0	0	0	3
To become familiar with simple synthetic illumination models and fundamental techniques for rendering synthetic images from 3Dscenes as well as to be exposed to aspects related to movement within these scenes.	3	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Victor Zordan – 06/15/06

COURSE DESCRIPTION

Dept., Number	CS133	Course Title	Computational Geometry
Units	4	Course Coordinator	Dimitrios Gunopulos
Required/elective	Elective	URL (if any):	http://www.cs.ucr.edu/~dg/cs133.html

Current Catalog Description

Introduction to the design of geometry algorithms. Covers the basic computational geometry concepts and techniques used in graphics, robotics, and engineering design. Topics include polygons and polytops, convex hulls, and voronoi diagrams.

Textbook

Computational Geometry in C (second edition) by Joseph O'Rourke, Cambridge University Press 1998

References/Materials

Course Goals/Objectives

1: To become familiar with fundamental concepts in computational geometry, including convex hulls, Voronoi diagrams, Delaunay triangulations, and other.
2: To learn the techniques for building robust geometric algorithms using efficient data structures, and to become proficient in implementing them.
3: To understand important applications of computational geometry in computer graphics, information systems, data mining, robotics, and other.
4: To learn how to model real life computational tasks using paradigms from computational geometry.

Prerequisites by Courses and Topics

CS 141, MATH 113, or equivalents.

Major Topics Covered in the Course

Visibility and Art Gallery Theorems (2 weeks)
Polygon Triangulation (2 weeks)
Convex Hulls in Two Dimensions (2 weeks)
Voronoi Diagrams(2 weeks)
Arrangements (1 week)
Search and Intersection (1 week)

Laboratory schedule: number of sessions per week and duration of each session

Class: 3 hours per week. Lab: One three hour laboratory session per week

Labs 1 and 2: introduction to computational geometry concepts
Labs 3 and 4: introduction to OpenGl
Labs 5, 6 and 7: working on the first project
Labs 8, 9and 10: working on the second project

Laboratory projects (specify number of weeks on each)

2 projects:
Project 1: implement, test, and visualize the results of a triangulation algorithm (3 weeks)
Project 2: understand a computational geometry problem, design an algorithm to solve it, theoretically derive the asymptotic running time of the solution, implement the solution, and provide a systems to visualize the results (3 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	30%	30%	Data Structures		15%
Software Design	15%	10%	Prog. Languages		
Comp. Arch.					

Oral and Written Communications:

Every student is required to submit at least ___2__ written reports (not including exams, tests, quizzes, or commented programs) of typically __2-3__ pages and to make __0__ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Algorithms and and analysis for Visibility problems in two dimensional polygons (Art Gallery Theorems) (2 weeks)
Algorithms for Polygon Triangulation, Trapezoidation and Partitioning into convex pieces (2 weeks)
Algorithms and lower bound analysis for computing Convex Hulls in two dimensions (2 weeks)
Algorithms and applications for Voronoi Diagrams and Delaunay Triangulations (2 weeks)
Algorithms and lower bounbs for Arrangements of lines in two dimensions (1 week)
Search and Intersection (1 week)

Problem Analysis

Please describe the analysis experiences common to all course sections.

The students learn about advanced algorithmic concepts using geometric problems. Geometric problems are intuitive and easy to understand, yet admit sophisticated and complex solutions. The students learn to design solutions to such problems, evaluate their solutions theoretically, and implement their solutions and visualize their solution.

Solution Design

Please describe the design experiences common to all course sections.

The students have to design an efficient algorithm that solves the problems described in the project descriptions, prove the efficiency of their solution, and implement their solution.

Assessment methods

· Assignments 35% (2 assignments with a programming and a written part each)

· Quizzes 30% (3 quizzes, the lowest grade is dropped)

· Final 35%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The students learn about advanced algorithmic concepts using geometric problems. Geometric problems are intuitive and easy to understand, yet admit sophisticated and complex solutions. The students design and implement solutions to such problems. The students learn to evaluate their solutions both theoretically and experimentally, and how to support their design choices by writing a report for each project (in addition to the program).

Relationship of course to program outcomes: The contribution of CS133 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
To become familiar with fundamental concepts in computational geometry including convex hulls Voronoi diagrams Delaunay triangulations and other.	3	0	0	0	0	0	0	0	0	0	3
To learn the techniques for building robust geometric algorithms using efficient data structures and to become proficient in implementing them.	3	0	1	0	0	0	0	0	0	0	3
To understand important applications of computational geometry in computer graphics information systems data mining robotics and other.	3	0	1	0	1	0	0	0	0	0	3
To learn how to model real life computational tasks using paradigms from computational geometry.	3	0	1	0	1	0	0	0	0	0	3

Prepared by, and date of preparation:
Dimitrios Gunopulos – 06/15/06

COURSE DESCRIPTION

Dept., Number	cs134	Course Title	Video Game Development
Units	4	Course Coordinator	V. Zordan
Required/elective	elective	URL (if any):	www.cs.ucr.edu/vbz/cs134.html

Current Catalog Description

Covers academic, theoretical, and practical aspects of video games by exploring common algorithms, data structures, and software design for different genres. Topics include game interface, character movement, intelligent behaviors, and networked or multiplayer games. Requires in-depth, applied programming and a term project, including the design, implementation, and analysis of a computer game.

Textbook

Intro to Game Development by S. Rabin

References/Materials

Core Techniques and algorithms in Game Programming by D. Sanchez-Crespo Dalmau
Various handouts

Course Goals/Objectives

An understanding of the theoretical aspects of video games.
An understanding of the key algorithms and data structures for video games.
Experience in the design and implementation of complex graphical user interfaces.
Additional experience in object-oriented design and analysis.
Additional experience in teamwork.

Prerequisites by Courses and Topics

CS 130 – Computer Graphics

Major Topics Covered in the Course

Game Design, Game Programming, AI for games, game interfaces, sound design, network, mobile gaming, serious games, industry production, architecture for game engines

Laboratory schedule: number of sessions per week and duration of each session

1 x 3 hrs /week

Laboratory projects (specify number of weeks on each)

game engine demo, scene graphs, interfaces, svn, sound, second life, work on projects

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	25%		Data Structures	10%	
Software Design	30%		Prog. Languages	10%	
Comp. Arch.	10%		*game related*	15%	

Oral and Written Communications:

Every student is required to submit at least __2___ written reports (not including exams, tests, quizzes, or commented programs) of typically __4___ pages and to make _____ oral presentations of typically __15__ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Social impact of games, and the effect of violence. Discussed, not graded.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Nature of play, social interactions afforded by games, games as tools for learning. 15%

Problem Analysis

Please describe the analysis experiences common to all course sections.

A small portion of lecture each week set aside to critically analyze various game designs

Solution Design

Please describe the design experiences common to all course sections.

Labs set up to have individual problems each lab session, must solve.

Assessment methods

written and oral progress reports, labs, final implementation of a game with team, team self and instructor based assessment.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Must work in teams, must be able to follow instructions, has a large coding development component and must be able to design software to manage complexity.

Relationship of course to program outcomes: The contribution of CS152 to program outcomes (a)-(k) or (1) – (3) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
An understanding of the theoretical aspects of video games.	0	0	0	0	0	0	0	3	3	3	0
An understanding of the key algorithms and data structures for video games.	2	0	0	0	2	0	0	0	0	0	0
Experience in the design and implementation of complex graphical user interfaces.	2	0	3	0	0	3	0	0	0	0	3
Additional experience in object-oriented analysis, design and programming.	2	2	3	0	3	0	0	0	0	0	3
Additional experience in teamwork.	0	2	0	3	0	3	3	0	0	0	3

Prepared by, and date of preparation:
Victor Zordan, 6/26/06

COURSE DESCRIPTION

Dept., Number	CS 141	Course Title	Intermediate Data Structures and Algorithms
Units	4	Course Coordinator	Stefano Lonardi, Neal Young
Required/elective	required	URL (if any):	

Current Catalog Description

Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming.

Textbook

"Introduction to the Design and Analysis of Algorithms" by A.V. Levitin, Addison Wesley.

References/Materials

"Algorithm Design (Foundations, Analysis, and Internet Examples)" by Michael T. Goodrich and Roberto Tamassia, Wiley.

"Introduction to Algorithms (2nd Edition)" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein, MIT Press.

Course Goals/Objectives

1: Perform asymptotic analysis of the efficiency of algorithms
2: Understand fundamental algorithms and data structures for discrete objects
3: Devise correct and efficient algorithms based on standard algorithmic design methods
4: Develop skills in systematic and rigorous computer programming by integrating the theory of algorithms with practical problem solving

Prerequisites by Courses and Topics

CS 014 with a grade of "C-" or better; CS 111/MATH 111; MATH 009C or MATH 09HC; proficiency in C++.

Major Topics Covered in the Course

RSA (modular arithmetic, greatest common divisor, Euclid's algorithm, extended Euclid's algorithm, multiplicative inverse, fast exponentiation, RSA public keys, private keys, encryption, decryption), divide and conquer (recurrence relations, geometric sums, naive upper and lower bounds on sums, large integer multiplication, polynomial multiplication, closest pair of points in a set), greedy algorithms (making change, Huffman coding, minimum spanning trees), dynamic programming (computing binomial coefficients, making change, optimal binary search trees, paragraph formatting), graphs (graph data structures, depth-first search, breadth-first search, directed dfs, topological sort finding cut vertices, Dijsktra's algorithm, Prim's MST algorithm), amortized data structures (union-find, growable array), lower bounds, NP-hardness

Laboratory schedule: number of sessions per week and duration of each session

Two hours of lab per week (one session)

Laboratory projects (specify number of weeks on each)

In each lab students are expected to complete a short self-contained project.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	100%		Data Structures		
Software Design			Prog. Languages		
Comp. Arch.					

Oral and Written Communications:

Every student is required to submit 0 written reports (not including exams, tests, quizzes, or commented programs) of typically 0 pages and to make 0 oral presentations of typically 0 minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

NONE

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Analysis of algorithms
Correctness of algorithms
Graph algorithms

Problem Analysis

Please describe the analysis experiences common to all course sections.

Design of algorithms, with emphasis on efficiency and correctness
Design principles (greedy, divide and conquer, dynamic programming)
Implementation of algorithms in C++

Solution Design

Please describe the design experiences common to all course sections.

Short self-contained lab projects
Written pencil and paper homeworks on algorithm analysis and design
Individual C++ projects

Assessment methods

Quizzes 35%, Final 35%, Homeworks 10%, Projects 10%, Laboratory 10%.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Relationship of course to program outcomes: The contribution of CS141 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Perform asymptotic analysis of the efficiency of algorithms	3	0	0	0	0	0	0	0	0	0	3
Understand fundamental algorithms and data structures fordiscrete objects	2	0	0	0	0	0	0	0	0	0	3
Devise correct and efficient algorithms based on standard algorithmic design methods	3	0	2	0	1	0	0	0	0	0	3
Develop skills in systematic and rigorous computer programming by integrating the theory of algorithms with practical problem solving	3	1	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Neal Young and Stefano Lonardi – 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 150	Course Title	The Theory of Automata and Formal Languages
Units	4	Course Coordinator	Marek Chrobak
Required/elective	required	URL (if any):	

Current Catalog Description:
Textbook: . Peter Linz, *An Introduction to Formal Languages and Automata*, Jones and Bartlett Publishers

References/Materials

Lecture notes.

Course Goals/Objectives:

Prerequisites by Courses and Topics:

Major Topics Covered in the Course: Finite automata: introduction, applications, regular languages, nondeterministic automata, equivalence and minimization of finite automata, string searching, regular expressions, regular grammars, closure properties, pumping lemma.

Context-free grammars: derivations, context-free languages, derivation trees, applications, normal forms, membership algorithm, pushdown automata, pumping lemma, closure properties, decision properties, parsing, LL(k) grammars

Introduction to Turing machines: Turing machine model of computation, undecidability.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; discussion, 1 hour.

Laboratory projects (specify number of weeks on each)

None.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	25%		Data Structures		
Software Design			Prog. Languages		
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least ___0__ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

N/A

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Mathematical theory of finite automata, context-free grammars 60%
Modeling real-life systems using finite-state machines 10%
Applications to string processing and parsing 20%
Computability 10%

Problem Analysis
Please describe the analysis experiences common to all course sections.

Students model real-life problems using finite-state machines. They analyze finite state machines and context-free grammars, by determining the languages recognized by them.

Solution Design

Please describe the design experiences common to all course sections.

Students need to design finite-state machines that perform specified tasks. They also design context-free grammars for specified languages.

Assessment methods: Quizzes 40%, Final 40%, Homeworks 20%.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

N/A

Relationship of course to program outcomes: The contribution of CS111 to program outcomes (a)-(k) or (1) – (7) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
To learn how to use correct mathematical terminology and notation.	3	0	0	0	0	0	0	0	0	0	3
To learn the methods of formal mathematical reasoning and proof techniques, including proofs by contradiction and by induction.	3	0	0	0	0	0	0	0	0	0	3
To learn how to model real-life problems using discrete mathematical structures: sets, sequences, combinations, permutations, graphs, trees, relations, and algebraic structures.	2	0	0	0	0	0	0	0	0	0	3
To master the concept of asymptotic notation and its application to estimating running time of various algorithms.	3	0	0	0	0	0	0	0	0	0	3
To learn fundamentals of number theory and its applications to cryptographic protocols.	2	0	0	0	0	0	0	0	0	0	3
To learn techniques for solving recurrence equations, and their applications to counting and to analyzing the complexity of divide-and-counter algorithms.	2	0	0	0	0	0	0	0	0	0	3
To learn the basic concepts in graph theory, including	1	0	0	0	0	0	0	0	0	0	3

connectivity, cycles, planarity, coloring.											

Prepared by, and date of preparation: Marek Chrobak, June 15 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 152	Course Title	Compiler Design
Units	4	Course Coordinator	Tom Payne
Required/elective	required	URL (if any):	

Current Catalog Description: Covers the fundamentals of compiler design, including lexical analysis, parsing, semantic analysis, compile-time memory organization, run-time memory organization, code generation, and compiler portability issues. Laboratory work involves exercises covering various aspects of compilers.

Textbook: *Modern Compiler Implementation in Java* (second edition) by Andrew Appel and Jens Palsberg

References/Materials

The web site for the text, http://www.cs.princeton.edu/~appel/modern/java/. Also, when Tom Payne teaches the class, the first seven chapters (75 pages) of his notes on Compiler Design, which he distribute to the students in postscript, dvi, and html formats.

Course Goals/Objectives:

- Provide students with a basic understanding of the design and functionality provided by compilers and interpreters, including theoretical foundations as far as necessary
- Provide students with practical experience building a compiler for a (small) imperative programming language, ideally generating code for an actual machine
- Using compilers, a well-explored field from the perspective of software engineering, illustrate various useful design and implementation techniques, focusing on object-oriented ones

Prerequisites by Courses and Topics: CS 061: Machine Organization and Assembly Language Programming; CS 141: Intermediate Data Structures and Algorithms; CS 150: Theory of Automata and Formal Languages

Major Topics Covered in the Course

29. Automatic generators for lexical analyzers (scanners): review of the set-of-states construction for determinizing nondeterministic finite automata, use of ordered EBNF to describe lexical categories, lookahead, left-context, case-study LEX, compression of state table.
30. Automatic generators for LALR parsers: Converting context-free grammars (BNF) to ``railroad diagrams'' to nondeterministic PDAs. Determinizing NPDAs: LR(0) tables, LR(1) tables, generating LALR tables from LR(1) table, LALR table by identifying states during construction of LR(1) table and propagation of lookaheads.
31. Syntax-directed translation.
32. The run-time environments including allocation and accessing of static, dynamic and automatic objects.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours

Laboratory projects (specify number of weeks on each)

One week of orientation to the term project, which is to construct a compiler for the MiniJava language specified at the back of the textbook.
One week on the use of LEX (FLEX) and generating scanner for term project.
One week on the use of YACC (BISON) and generating parser for term project.
One week on the building of the syntax tree for the term project.
Two weeks on semantic analysis and checking for the term project.
Three weeks on code generation for the term project.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5.00%		Data Structures	5.00%	
Software Design	24.00%		Prog. Languages	33.00%	33.00%
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically __0___ pages and to make ___0__ oral presentations of typically ___0__ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

They are told the importance of doing their own work and fail the course if they violate that rule

The are taught the golden rule with respect to error messages, and are shown easy ways to make error messages more helpful. The only feedback on that is in the form of advice from the TAs during labs.

The students are lectured on the importance of high-level programming languages to programmer productivity and the importance of programmer productivity to all of information technology.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

This course is where the material such as context free grammars and syntax trees is put to practical use in building important tools. Specifically, the set-of-states construction learned in the theory of computation class is used in the construction of scanner generators and parser generators. (That construction is covered only in class. Then the students use such generators in the development of the scanner and parser for their term project.)

Problem Analysis

Please describe the analysis experiences common to all course sections.

The analysis and design for the overall framework for the term project is presented in class and follows the presentation in the text. The analysis and design for some of the components (semantic analyzers and code generators for the individual expressions, declarations, and statements) are presented in class, but the students must do the rest on their own, with some in-lab help from the TAs.

Solution Design

Please describe the design experiences common to all course sections.

Part of the description of design experience is given in the box above. Beyond that, we tend to emphasize object-oriented, test-driven design. We recommend that students treat each type of node in the syntax tree as a class, with the various kinds of expressions being subclasses of the Expression class, etc. Each of those classes must have a constructor, a method for doing semantic analysis, and a method for generating code. So, each class become a nicely constrained design problem.

Assessment methods: Final exam: 34%; Project: 33%; Quizzes: 33%.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The design and implementation of a compiler is a significant design experience and teaches a lot about good coding and testing practices. But in CS152 the training wheels are on. Students are presented with much of the top-level analysis and design, and they are coached a lot on how to stay out of trouble. This is realistic preparation for the capstone design courses, in which the training wheels are off.

Relationship of course to program outcomes: The contribution of CS152 to program outcomes (a)-(k) or (1) – (3) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Provide students with a basic understanding of the design and functionality provided by compilers and interpreters, including theoretical foundations as far as necessary	1	0	0	0	0	0	0	0	0	0	3
Provide students with practical experience building a compiler for a (small) imperative programming language, ideally generating code for an actual machine	3	0	3	0	1	0	0	0	0	0	3
Using compilers, a well-explored field from the perspective of software engineering, illustrate various useful design and implementation techniques, focusing on object-oriented ones	3	0	2	0	1	0	0	0	0	0	3

Prepared by, and date of preparation:
Tom Payne 6/18/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 153	Course Title	Design of Operating Systems
Units	4	Course Coordinator	Vana Kalogeraki
Required/elective	Required	URL (if any):	http://www.cs.ucr.edu/~vana/cs153

Current Catalog Description

Principles and practice of operating system design, including concurrency, memory management, file systems, protection, security, command languages, scheduling, and system performance. Laboratory work involves exercises covering various aspects of operating systems

Textbook

Operating System Concepts by Avi Silberschatz, Peter Baer Galvin, Greg Gagne, John Wiley & Sons, Sixth Edition
Kernel Projects for Linux by Gary Nutt, Addison Wesley

References/Materials

http://www.cs.ucr.edu/~vana/cs153/resources.htm Additional resources for the course
http://www.cs.ucr.edu/~vana/cs153/assignments.htm Supporting material for the project assignments

Course Goals/Objectives

- Study basic principles underlying the design of operating systems with a focus on principles and mechanisms used throughout the design
- An understanding of CPU scheduling, storage management: memory management, virtual memory and file systems
- Study of concurrency control and synchronization, classic algorithms for synchronization and concurrency management
- Study Deadlocks Devices, device management and I/O systems
- Study dynamic binding
- An understanding of protection, access control and security
- Improve skills in concurrent programming and introduce kernel programming

Prerequisites by Courses and Topics

CS 061. Machine Organization and Assembly Language Programming
CS 141. Intermediate Data Structures and Algorithms
C++ programming proficiency

Major Topics Covered in the Course

Introduction to Operating Systems, Computer-System Structures and Operating-System Structures. Process Management: Processes, Threads, CPU Scheduling, Process Synchronization and Deadlocks. Storage Management: Memory Management, Virtual Memory, File-System Interface, File-System Implementation.

Laboratory schedule: number of sessions per week and duration of each session

Lecture 3 hours
Laboratory 3 hours

Lab1: Practicing system functions
Lab2: Working on the 1st project assignment - shell programming
Lab3: Working on the 1st project assignment - shell programming
Lab4: Introduction to multi-threading
Lab5: Working on the 2nd project assignment – multithreading
Lab6: Working on the 2nd project assignment – multithreading
Lab7: Introduction to file systems
Lab8: Working on the 3rd project assignment – file systems
Lab9: Working on the 3rd project assignment – file systems
Lab10: Working on the 3rd project assignment – file systems

Laboratory projects (specify number of weeks on each)

1st project – Shell programming: 2 weeks
2nd project – Multithreading programs: 2 weeks
3rd project – File system: 3 weeks

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	10	Data Structures	10	5
Software Design	20	15	Prog. Languages	15	5
Comp. Arch.	5	5			

Oral and Written Communications:

Every student is required to submit at least __1__ written reports (not including exams, tests, quizzes, or commented programs) of typically _2-3__ pages and to make __1__ oral presentations of typically __10_ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students will learn the importance of social and ethnical issues by participating in teams and working with other team members.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Introduction to Operating Systems Structures – 1 lecture
Introduction to Computer System Structures – 1 lecture
Process Management – 2 lectures
Threads – 2 lecture
CPU Scheduling - 2 lectures
Process Synchronization – 2 lectures
Deadlocks – 3 lectures
Memory Management – 3 lectures
Virtual Memory – 2 lectures
File systems – 2 lectures

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions.
The students will learn the importance of analysis through hands-on experience.

Solution Design

Please describe the design experiences common to all course sections.

A primary goal of the course is the design of solutions that meet the project requirements. The students will design, implement and test various components of an operating system (a UNIX Shell, a CPU Scheduler, a Multithreading program and a File System). The students will apply software tools to build and evaluate their designs.
The students will learn the importance of solution design through hands-on experience. This will help them understand the complexity of building real systems.

Assessment methods

Homeworks – 10%
Projects – 30%
2 Midterm Exams – 20%
Final Exam – 40%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students will learn how to design and implement various components of an operating system, thus they will understand the overall engineering design process (identification of constraints, description of design criteria and objectives, usage of tools, development of a prototype, evaluation of the prototype based on the design criteria). Students will get hands-on experience in proposing, designing and executing the project.
Students will also learn how to work in teams; each student will actively participate as a member of a team, collaborate with the other team members, share the daily design activities and management of the project and contribute to achieve the project goals. At the end of the project, they will have to write a team report that describes their design, thus they will learn how to communicate their ideas effectively.

Relationship of course to program outcomes: The contribution of CS153 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Study basic principles underlying the design of operating systems with a focus on principles and mechanisms used throughout the design	1	0	1	0	1	0	0	0	0	0	3
An understanding of CPU scheduling storage management: memory management virtual memory and file systems	1	0	0	0	0	0	0	0	0	0	3
Study of concurrency control and synchronization classical algorithms for synchronization and concurrency management	1	0	1	0	0	0	0	0	0	0	3
Study Deadlocks Devices device management and I/O systems	1	0	0	0	0	0	0	0	0	0	3
Study dynamic binding	0	0	0	0	0	0	0	0	0	0	3
An understanding of protection access control and security	1	0	0	0	0	1	0	1	0	0	3
Improve skills in concurrent programming and introduce kernel programming	3	2	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Vana Kalogeraki – 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 160	Course Title	Concurrent Programming and Parallel Systems
Units	4	Course Coordinator	Tom Payne
Required/elective	elective	URL (if any):	

Current Catalog Description: Study of concurrent and parallel systems. Topics include modular structure and design, inter-process communication, synchronization, failures and persistence, concurrency control, atomic transactions, recovery, language support, distributed inter-process communication, and implementation mechanisms. Provides preparation for the study of operating systems, databases, and computer networking.

Textbook: Concurrent Systems 2nd Ed, Jean Bacon

References/Materials

Selected handouts.

Course Goals/Objectives:

- Understand the requirements to support concurrent systems.
- Introduce modular system structure for concurrent systems and the relationship to processes and threads.
- Understand the process abstraction, support for concurrency, and dynamic execution models.
- Understand the difference between process abstraction versus dynamic execution models that share and address space.
- Understand process interaction, hardware support for process interaction, concurrency control without hardware support, and semaphores.
- Introduce classic systems problems and the POSIX threads package.
- Introduce IPC mechanisms for shared memory and non-shared memory systems.
- Understand mechanisms used to support crash resilience and introduce persistent data.
- Understand composite operations that span distributed systems in the presence of concurrency and crashes and the fundamentals of transactions.
- Introduce concurrency control for transactions.
- Provide laboratories that improve student programming competence and train students to better design, implement and analyze concurrent systems.
- Provide assignments that give substantial hands on experience writing systems that use concurrency and require concurrency control and fine grain concurrency support.

Prerequisites by Courses and Topics: CS 061: Machine Organization and Assembly Language Programming; CS 141: Intermediate Data Structures and Algorithms.

Major Topics Covered in the Course

Processes, threads, locks, semaphores, monitors, condition variables, interrupts/signals, commit protocols, transactions, IPC, timeouts.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		20.00%	Data Structures		10.00%
Software Design		30.00%	Prog. Languages		25.00%
Comp. Arch.		15.00%			

Oral and Written Communications:
Every student is required to submit at least _____ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Dekker's algorithm, Lamport's bakery algorithm, dining philosophers problem, two-phase commit protocol.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Different homework problems are given for each offering. There is the standard analysis that is needed to understand a frame problems in concurrent programming. There is also the problem of building an understanding of both the problem and the tool set to be used in solving it.

Solution Design

Please describe the design experiences common to all course sections.

The solutions involve designing, implementing, and testing software systems of modest size.

Assessment methods: 40% homework, 30% quizzes, 30% final exam.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The assignments involve typical problems in small system design and in the use of standard tools, e.g., the Pthreads system, which is the major standard in C-based multithreaded programming.

Relationship of course to program outcomes: The contribution of CS160 to program outcomes (a)-(k) or (1) – (12) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand the requirements to support concurrent systems.	1	0	1	0	0	0	0	0	0	0	3
Introduce modular system structure for concurrent systems and the relationship to processes and threads.	1	0	0	0	0	1	0	0	0	0	3
Understand the process abstraction, support for concurrency, and dynamic execution models.	2	0	0	0	0	0	0	0	0	0	3
Understand the difference between process abstraction versus dynamic execution models that share and address space.	1	0	0	0	0	0	0	0	0	0	3
Understand process interaction, hardware support for process interaction, concurrency control without hardware support, and semaphores.	1	0	0	0	0	0	0	0	0	0	3
Introduce classic systems problems and the POSIX threads package.	0	0	0	0	0	0	0	0	0	0	3
Introduce IPC mechanisms for shared memory and non-shared memory systems.	1	0	0	0	0	0	0	0	0	0	3
Understand mechanisms used to support crash resilience and introduce persistent data.	1	0	0	0	0	0	0	0	0	0	3
Understand composite operations that span distributed systems in the presence of concurrency and crashes and the fundamentals of transactions.	1	0	0	0	0	0	1	0	0	0	3
Introduce concurrency control for transactions.	1	0	0	0	0	0	0	0	0	0	3

Provide laboratories that improve student programming competence and train students to better design, implement and analyze concurrent systems.	3	2	3	0	2	0	0	0	0	0	3
Provide assignments that give substantial hands on experience writing systems that use concurrency and require concurrency control and fine grain concurrency support.	3	0	2	0	2	0	0	0	0	0	3

Prepared by, and date of preparation: Tom Payne 6/20/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 161	Course Title	Design and Architecture of Computer Systems
Units	4	Course Coordinator	Dr. Laxmi N. Bhuyan
Required/elective	Elective	URL (if any):	http://www.cs.ucr.edu/~bhuyan/cs161 (lectures) http://www.cs.ucr.edu/~vladimir/cs161 (discussions)

Current Catalog Description: A study of the fundamentals of computer design. Topics include the performance evaluation of microprocessors, instruction set design and measurements of use, microprocessor implementation techniques including multi-cycle and pipelined implementations, computer arithmetic, memory hierarchy, and input/output (I/O) systems.

Textbook: Patterson and Hennessy, "Computer Organization and Design" Morgan Kaufmann publisher

References/Materials

Lecture slides, available via the lectures web site; selected discussion notes, available via the discussions web site.

Course Goals/Objectives:

1. Understand instructions as the language of the machine and the tradeoffs in instruction set design
2. Introduction to the issues and factors that impact performance, both hardware and software
3. Learn how to design the data-path and control unit as the heart of the CPU
4. Introduction to computer arithmetic: fast addition and multiplication
5. Introduction to memory hierarchy: simple caches and virtual memory
6. Learn how to design fast CPUs using pipelining
7. Introduction to advanced processors using instruction level parallelism

Prerequisites by Courses and Topics: CS 120B/EE 120B: Introduction to Embedded Systems. Introduction to hardware and software design of digital computing systems embedded in electronic devices (such as digital cameras or portable video games). Topics include custom and programmable processor design, standard peripherals, memories, interfacing, and hardware/software tradeoffs. Laboratory involves use of synthesis tools, programmable logic, and microcontrollers and development of working embedded systems; concurrent enrollment in CS 161L.

Major Topics Covered in the Course

Chapter 1: Introduction
Chapter 2: MIPS Instructions
Chapter 3: Computer Arithmetic
Chapter 4: Understanding Performance
Chapter 5: The Processor: Datapath and Control
Chapter 7: Memory Hierarchy
Chapter 8: I/O System

Laboratory schedule: number of sessions per week and duration of each session:

There is a separate laboratory class, CS 161 L, which is a co-requisite for the course. In addition to the 3 hours a week of lectures, this course has 1 weekly discussion session, which lasts for 50 minutes.

Weekly Discussion Schedule:

Week 1: Introduction
Week 2: Homework 1 (**Assessing and Understanding Performance**) introduced
Week 3: Solutions for Homework 1, Homework 2 (**Instructions**) introduced
Week 4: Homework 2 discussed
Week 5: Solutions for Homework 2, Homework 3 (**Arithmetic**) introduced
Week 6: Homework 3 discussed
Week 7: Solutions for Homework 3, Homework 4 (**Datapath and Control**) introduced
Week 8: Homework 4 discussed
Week 9: Solutions for Homework 4, Homework 5 (**Memory Hierarchy**) introduced
Week 10: Overview, Solutions for Homework 5

Laboratory projects (specify number of weeks on each)

There were no specific projects for the discussion sessions.

Instead, there were 5 homeworks, and discussions were structured to present material and strategies useful for solving the homeworks. Solutions to selected problems were presented after homeworks have been graded.

On average, for each of the 5 homework topics two weeks were allocated:

- Assessing and Understanding Performance (2 weeks)
- Instructions (2 weeks)
- Arithmetic (2 weeks)
- Datapath and Control (2 weeks)
- Memory Hierarchy (2 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced

Algorithms			Data Structures		
Software Design			Prog. Languages		
Comp. Arch.	100				

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically _0____ pages and to make __0___ oral presentations of typically _0____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students work in groups during the discussion session. They get a chance to interact with each other. They are constantly lectured about the benefit of graduate studies.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

The theoretical material consists of understanding of the instruction sets of a typical CPU, basic building blocks of datapath, and memory units. Approximately 50% of the time is spent in explaining these operations.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analyzing the problem to get the understanding of the requirements was emphasized in the lectures and in the discussions.

Solution Design

Please describe the design experiences common to all course sections.

This course emphasizes quantitative approach to computer architecture; hence, a significant number of problems required quantitatively evaluating several alternative solutions to find the best one for the problem at hand.

Assessment methods:

20% Homeworks (5 homeworks, 4% each)

20% Exam 1

25% Exam 2

35% Exam 3

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Relationship of course to program outcomes: The contribution of CS161 to program outcomes (a)-(k) or (1) – (7) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand instructions as the language of the machine and the tradeoffs in instruction set design	1	1	1	0	1	0	0	0	0	0	3
Introduction to the issues and factors that impact performance, both hardware and software	2	2	2	0	1	0	0	0	0	0	3
Learn how to design the data-path and control unit as the heart of the CPU	3	2	3	0	0	0	0	0	0	0	3
Introduction to computer arithmetic: fast addition and multiplication	3	0	3	0	2	0	0	0	0	0	3
Introduction to memory hierarchy: simple caches and virtual memory	2	2	3	0	2	0	0	0	0	0	3

Prepared by, and date of preparation:
L.N. Bhuyan, June 22, 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 161L	Course Title	Laboratory in Design and Architecture of Computer Systems
Units	4	Course Coordinator	Walid Najjar
Required/elective	required	URL (if any):	

Current Catalog Description: Students design and simulate a complete computer system, using hardware description language and simulator. Topics include instruction set architecture design, assemblers, data-path and control unit design, arithmetic and logic unit, memory and input/output (I/O) systems, and integration of all parts into a working computer system.

Textbook: *Computer Organization and Design: The HW / SW Interface*, David A Patterson, John L Hennessy

References/Materials

Course web site on moodle http://fish.cs.ucr.edu/moodle/

Course Goals/Objectives:

- Understanding of computer arithmetic by (1) Design and implementation of an ALU and (2) Implementation of complex arithmetic algorithms in software.

- Understanding of operation of a CPU by (1) Design and implementation of a data-path and (2) Design and implementation of the control unit both for the MIPS architecture.

- Understanding of operation of a cache memory by designing and writing a cache-simulator program in C/C++.

- Familiarity with the cycle-level simulation of a complex computer architectures

- Understanding of data-paths via hands on introduction to data-paths.

Prerequisites by Courses and Topics: CS 120B/EE 120B: Introduction to Embedded Systems. Introduction to hardware and software design of digital computing systems embedded in electronic devices (such as digital cameras or portable video games). Topics include custom and programmable processor design, standard peripherals, memories, interfacing, and hardware/software tradeoffs. Laboratory involves use of synthesis tools, programmable logic, and microcontrollers and development of working embedded systems; concurrent enrollment in CS 161.

Major Topics Covered in the Course

Review of VHDL, implementation of adders, implementation of 16/32-bit ALUs, floating-point arithmetic, microprocessor data-paths, complete microprocessor implementation (with external memory and control unit), cache simulator.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture 3 hours
Laboratory 3 hours

Lab1: Review of VHDL, implementation of adders.
Lab2: Implementation of 16/32-bit ALU – part 1.
Lab3: Implementation of 16/32-bit ALU – part 2.
Lab4: Floating-point arithmetic and ALU – part 1.
Lab5: Floating-point arithmetic and ALU – part 2.
Lab6: A microprocessor data-path.
Lab7: Completed microprocessor implementation (external memory and control unit).
Lab8: Completed microprocessor implementation – part 2.
Lab9: Cache simulator – part 1.
Lab10: Cache simulator – part 2.

Laboratory projects (specify number of weeks on each)

1st project – Integer ALU: 2 weeks.
2nd project –Floating-point ALU: 2 weeks.
3rd project – Microprocessor implementation: 3 weeks.
4th project – Cache simulator: 2 weeks.

Estimate Curriculum Category Content (percent of time)

Area	**Core**	**Advanced**	**Area**	**Core**	**Advanced**
Algorithms	10	10	Data Structures	5	5
Software Design	15	15	Prog. Languages	10	10
Comp. Arch.	60	60			

Oral and Written Communications:
For each project each team is required to submit at least __1__ written reports (not including exams, tests, quizzes, or commented programs) of typically _2-3__ pages and to make __1__ oral presentations of typically __10_ minute's duration.

Social and Ethical Issues

Students will learn the importance of social and ethnical issues by participating in teams and working with other team members.
Topic of social, ethical and legal issues in architecture design is addressed in the lecture.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Review of number representation, integer and floating-point – 1 lecture.
Integer addition, carry-ripple, carry-save and carry select – 2 lecture.
Integer multiplication (bit-recoding, array multipliers) – 2 lecture.
Fast integer division – 2 lecture.
Basic control unit design – 1 lecture.
Cache structure and design – 2 lectures.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions.
The students will learn the importance of analysis through hands-on experience.

Solution Design

Please describe the design experiences common to all course sections.

A primary goal of the course is to teach hands-on the design and implementation of a basic microprocessor starting with individual components and building a complete system.
An important aspect is the design relationship between the instruction set architecture and the system implementation and hence between software and hardware.
The students will learn the importance of solution design through hands-on experience. This will help them understand the complexity of building real systems.

Assessment methods

Projects – 80%
Quizes (3) – 20%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students will learn how to design and implement various components of a microprocessor system, thus they will understand the overall engineering design process (identification of constraints, description of design criteria and objectives, usage of tools, development of a prototype, evaluation of the prototype based on the design criteria). Students will get hands-on experience in proposing, designing and executing the project.
Students will also learn how to work in teams; each student will actively participate as a member of a team, collaborate with the other team members, share the daily design activities and management of the project and contribute to achieve the project goals. At the end of the project, they will have to write a team report that describes their design, thus they will learn how to communicate their ideas effectively.

Relationship of course to program outcomes: The contribution of CS161L to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understanding of computer arithmetic by (1) Design and implementation of an ALU and (2) Implementation of complex arithmetic algorithms in software.	3	0	3	0	1	0	0	0	0	0	3
Understanding of operation of a CPU by (1) Design and implementation of a data-path and (2) Design and implementation of a the control unit both for the MIPS architecture	3	0	3	0	1	0	0	0	0	0	3
Understanding of operation of a cache memory by designing and writing a cache-simulator program in C/C++	3	0	2	0	1	0	0	0	0	0	3
Familiarity with the cycle-level simulation of a complex computer architectures	1	0	0	0	0	0	0	0	0	0	3
Understanding of data-paths via a hands on introduction to data-paths	3	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:
Walid Najjar, 19 June 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 162	Course Title	Computer Architecture
Units	4	Course Coordinator	
Required/elective	elective	URL (if any):	

Current Catalog Description: The study of advanced processor design. Topics include CPU pipelining, data and control hazards, instruction-level parallelism, branch prediction, and dynamic scheduling of instructions. Also covers Very Long Instruction Word (VLIW) processing, design of network and embedded processors, basic multiprocessor design, shared memory and message passing, and network topologies.

Textbook:
1. Patterson and Hennessy, "Computer Organization and Design: The Hardware/Software Interface" Morgan Kaufmann
2. Hennessy and Patterson, "Computer Architecture: A Quantitative Approach" Morgan Kaufmann

References/Materials

Supporting material for the course are provided at http://www.cs.ucr.edu/~bhuyan/cs162/index.html and http://www.cs.ucr.edu/~junyang/teach/S06_162/162.html

Course Goals/Objectives:
1. Learn various techniques, such as instruction level parallelism, VLIW, multithreading, branch prediction and speculation, for design of advanced processors. Evaluate the performance impact of some of these techniques through simulations.
2. Learn architectural details of network processors. Develop network protocol applications and evaluate their performance on a network processor through simulation and experiment.
3. Learn basics of multiprocessor operation including shared memory and message passing communications, cache coherence protocols, and user level communication techniques.

Prerequisites by Courses and Topics: CS 161 and CS 161L: Design and Architecture of Computer Systems. A study of the fundamentals of computer design. Topics include the performance evaluation of microprocessors, instruction set design and measurements of use, microprocessor implementation techniques including multi-cycle and pipelined implementations, computer arithmetic, memory hierarchy, and input/output (I/O) systems.

Major Topics Covered in the Course:

- Advanced processor design: CPU pipelining, Datapath and Control Design, Data and Control Hazards: The topics will be covered from Chapter 6 of the text 1
- Instruction level parallelism, Dynamic scheduling of instructions, Branch Prediction and Speculation – From text book (2) and papers
- VLIW, Multithreading, and Network processor architectures – From papers
- Basic multiprocessor design: Shared memory and message passing; Network topologies. The topic will be covered from Chapter 9 of the text 1.

Laboratory schedule: number of sessions per week and duration of each session: The following laboratory assignments were given in spring 2004.

33. **SimpleScalar Simulation Tool Set**
34. **Performance Evaluation with Benchmarks (I)**
35. **Performance Evaluation with Benchmarks (II)**
36. **Intel IXP2400 Overview and Hands-on Practice**
37. **Microcode Programming on Intel IXP2400**
38. **Packet Receiving, Processing and Transmitting**
39. **Packet Processing in a Single Thread**
40. **Unordered Thread Execution**
41. **Context Pipeline Stages**
42. **Rings and Queues**

Homework:

 Homework 1: 6.17, 6.18 on pp. 456 of the textbook. Due: 4/27 before class.
 Homework 2: multicycle staged pipeline. Due: 5/9 before class.
 Homework 3: Tomasulo's algorithm. Assigned: 5/17/2006. Due: 5/25/2006, in class.
 Homework 4: Virtual memory. Assigned: 5/26. Due 6/1 in class.
 Homework 5: read the example on pp. 601 of the textbook. Recalculate the result using a memory backplane bus transfer rate of 500MB/sec, i.e., the original rate has been cut by half.

Laboratory projects (specify number of weeks on each)

Students are divided to groups of two students and complete one of the following projects in 2/3 weeks time.
1 TCP splicing: Offer content based switching by splicing two TCP connections
2 Device driver for IXP2400: Make IXP2400 talk to host over PCI-to-PCI bridge
3 Instruction decoder: write an instruction decoder for IXP2400
4 SSL offloading: offload SSL processing to IXP
5 Transcoding: Speed up media transcoding with IXP2400
6 IPv4 packet forwarding and classification
7 Performance evaluation of cryptographic primitives on IXP2400

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5	5	Data Structures		
Software Design	20	10	Prog. Languages		
Comp. Arch.	40	20			

Oral and Written Communications:
Every student is required to submit at least _one____ written reports (not including exams, tests, quizzes, or commented programs) of typically __10___ pages and to make __1___ oral presentations of typically __15___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students will learn the importance of social and ethnical issues by participating in teams and working with other team members.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

The course covers both theoretical and practical aspects of computer design. Theoretical aspects include learning various concepts in computer architecture and takes about 30% of the total time.

Problem Analysis

Please describe the analysis experiences common to all course sections.

When the course was started, emphasis was given to laboratory experiments consisting of network processors (NP). Since NP programming is done at an assembly language level, many students found it to be difficult to learn another language at their final year. Also, there was not enough time to learn all the techniques of NP programming during one quarter. Hence the NP laboratory project was removed when the course was offered in Spring 2006.

Solution Design

Please describe the design experiences common to all course sections.

Computer architecture is a subject full of hardware design. The students learn to design the datapath and control unit of various CPUs.

Assessment methods:

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

The course teaches students how to become a computer engineer concentrating on the electronic design of its components. They also learn how to work in teams and write reports.

Relationship of course to program outcomes: The contribution of CS162 to program outcomes (a)-(k) or (1) – (3) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Learn various techniques, such as instruction level parallelism, VLIW, multithreading, branch prediction and speculation, for design of advanced processors. Evaluate the performance impact of some of these techniques through simulations	3	2	0	0	0	0	0	0	0	0	3
Learn architectural details of network processors. Develop network protocol applications and evaluate their performance on a network processor through simulation and experiment	2	2	0	0	0	0	0	0	0	0	3
Learn basics of multiprocessor operation including shared memory and message passing communications, cache coherence protocols, and user level communication techniques	1	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation: Tom Payne, June 15 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 164	Course Title	Computer Networks
Units	4	Course Coordinator	Michalis Faloutsos
Required/elective	elective	URL (if any):	http://www.cs.ucr.edu/cs164/

Current Catalog Description: Covers the fundamentals of computer networks. Topics include layered network architecture, communication protocols, local area networks, UNIX network programming, verification, network security, and performance studies.

Textbook: L. L. Peterson and B. S. Davie, *Computer Networks*, *A Systems Approach*, Third Edition, Morgan Kaufmann, 2003.

References/Materials

- The Pocket Guide to TCP/IP Sockets by Michael J. Donahoo and Kenneth L. Calvert
- Unix Network Programming: Volume I by W.Richard Stevens Prentice Hall, 1998.
- Computer Networking: A Top-Down Approach Featuring the Internet by James Kurose and Keith Ross. Addison Wesley
- Data Networks by D.Bersekas and R.Gallager. Prentice Hall.

Course Goals/Objectives:

1. Understanding of layering and the network stack concept
2. Understanding of techniques for reliable transmission
3. Understanding of IP routing
4. Understanding of common protocols on all layers of the network stack
5. Experience with the BSD socket API
6. Ability to track the actions associated with transmitting an application-level datagram down through the network stack, through the network, and to a remote application.

Prerequisites by Courses and Topics: CS 141: Intermediate Data Structures and Algorithms. Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming; CS 153: Design of Operating Systems. Principles and practice of operating system design, including concurrency, memory management, file systems, protection, security, command languages, scheduling, and system performance. Laboratory work involves exercises covering various aspects of operating systems.

Major Topics Covered in the Course

- Week1: Introduction to Computer Networks
- Week2: Data Link Layer
- Week3: Routing
- Week4: Routing (cont.)
- Week5: Transport
- Week6: Transport (cont.)
- Week7: Transport (cont.)
- Week8: Network Security, Application Layer, Miscellaneous Issues
- Week9: Network Security, Application Layer, Miscellaneous Issues
- Week10: Network Security, Application Layer, Miscellaneous Issues

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

- Lab 1: (1 week) Use *traceroute* to study Internet packet forwarding to different continents. Modify the supplied *echo server* in application to become a remote arithmetic checker.
- Lab 2: (1 week) Write programs to play the roles of the *client*, *forwarder* and *server* in a simple Peer-to-Peer (P2P) network application..
- Lab 3: (1 week) Run a simple network simulation example using the "*ns*" network simulation package.
- Lab 4: (1 week) Develop a more complicated network simulation with "*ns*" that includes trace-driven input.
- Lab 5: (1 week) Experiment with "*distance vector*" and "*link state*" routing protocols in the presence of link failures, using "*ns*".
- Lab 6: (1 week) Use the "*ethereal*" packet-level monitoring tool (aka "sniffer") to study network traffic. For extra credit, reassemble the payloads of related packets to view web pages being downloaded over the monitored link.
- Lab 7: (1 week). Study the dynamic behavior of the TCP congestion avoidance algorithm using "*ns*".
- Lab 8: (1 week) Compare the dynamic behavior of three different versions of TCP on a multihop connection using "*ns*".
- Project: (3 weeks). Write a 4-6 page research paper as part of a two-person team.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	10	Data Structures		
Software Design		5	Prog. Languages		5
Comp. Arch.	10	10			

Oral and Written Communications:
Every student is required to submit at least ___1__ written reports (not including exams, tests, quizzes, or commented programs) of typically __6___ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Introduction to network security including information privacy, tampering by intermediate nodes in a multihop path; denial of service attacks, etc. It is covered in short-answer test questions. (1 lecture)

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Complexity of shortest-path algorithms in the context of network routing (1 lecture); spanning trees for local networks and multicast distribution (1 lecture).

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions. More specifically, students will learn the importance of analysis through hands-on experience. Calculation of the bandwidth-delay product for a communication link or network path, and its significance in determining the capacity of acknowledgement-based flow control schemes (1 lecture).

Solution Design

Please describe the design experiences common to all course sections.

Socket programming to provide asynchronous bi-direction exchange of information between processes running on different computers linked by a network. Understanding protocol descriptions expressed as event-driven finite state machines, and implementing a non-trivial protocol on a real network (via sockets) or simulated network.

Assessment methods:

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Understanding the importance of commonly used network standards for low-level physical connections (for example, IEEE 802 standards for Local and Metropolitan Area Networks) and higher level protocols for managing the network or application layer functions (for example, Internet Engineering Task Force RFCs). Understanding why certain technologies become dominant (for example, Ethernet and TCP/IP) while others fail (for example, Token Ring and ATM).

Relationship of course to program outcomes: The contribution of CS164 to program outcomes (a)-(k) or (1) – (6) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understanding of layering and the network stack concept	1	2	0	0	0	0	0	0	0	0	3
Understanding of techniques for reliable transmission	1	2	0	0	0	0	0	0	0	0	3
Understanding of IP routing	1	0	0	0	0	0	0	0	0	0	3
Understanding of common protocols on all layers of the network stack	1	0	0	0	0	0	0	0	0	0	3
Experience with the BSD socket API	2	0	0	0	0	0	0	0	0	0	3
Ability to track the actions associated with transmitting an application-level datagram down through the network stack, through the network, and to a remote application.	2	2	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation:

Mart Molle, June 19, 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 165	Course Title	Computer Security
Units	4	Course Coordinator	C.V. Ravishankar
Required/elective	elective	URL (if any):	

Current Catalog Description: Examines the ways in which information systems are vulnerable to security breaches. Topics include attacks; security labels, lattices, and policies; safeguards and countermeasures; intrusion detection; authorization and encryption techniques; networks; digital signatures, certificates, and passwords; privacy issues, firewalls, and spoofing; Trojan horses and computer viruses; CERT Coordination Center; and electronic commerce.

Textbook:

References/Materials

Textbook: Security in Computing by C.P.Pfleeger and S.L.Pfleeger, Prentice Hall, 3rd Edition.

Course Goals/Objectives:

43. Understand basic issues to be addressed secure computer systems design, including risk, privacy, integrity, and availability.
44. Understand the basic security and threat models.
45. Understand the principles of cryptography.
46. Understand various protocols for authentication, data security and privacy, key generation and management, and application-specific security.
47. Understand program security, including malicious code and countermeasures.
48. Understand the principles of secure systems design.
49. Understand the principles of network and distributed systems security.
50. Develop an appreciation of issues systems administration.
51. Write programs to reinforce concepts learned.

Prerequisites by Courses and Topics: CS 141: Intermediate Data Structures and Algorithms. Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming; CS 153: Design of Operating Systems. Principles and practice of operating system design, including concurrency, memory management, file systems, protection, security, command languages, scheduling, and system performance. Laboratory work involves exercises covering various aspects of operating systems.

Major Topics Covered in the Course

Introdcution to Computer Security, Elementary Cryptography (Both Secret Key and Public Key Systems), Program security, Viruses and Worms, Protection in General Purpose Operating Systems, Network Security.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

- Implementation of the DES Encryption and Decryption Algorithms. (3 weeks)
- Implementation of a Public Key Exchange system based on a Diffie Hellman exchange (3 weeks)
- Implementation of a Secure Socket Layer on a UDP connection. (4 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	25	25	Data Structures	10	10
Software Design	15		Prog. Languages	5	
Comp. Arch.	5	5			

Oral and Written Communications:
Every student is required to submit at least ___1__ written reports (not including exams, tests, quizzes, or commented programs) of typically _3____ pages and to make _1____ oral presentations of typically __15___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

The initial projects are individual projects. The final project is a team project. The students are advised of the ethical issues in terms of independently working on the initial projects and working together and contributing to the team effort.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Introduction to Computer Security 3 lectures.
Elementatry Cryptography – Secret Keys, DES and AES – 3 lectures
Public Key Infrastructure – 2 lectures
Viruses and Worms – 4 lectures
Protection in general purpose operating systems -- 4 lectures
Network security (Denial of Service attacks, Firewalls, Intrusion Detection Systems, Secure E-mail) – 4 lectures

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions.
The students will learn the importance of analysis through hands-on experience.

Solution Design

Please describe the design experiences common to all course sections.

The students learn to design solutions to various types of security threats. In particular, they learn to implement cryptographic modules and the design of a secure message exchange mechanism. The students will apply software tools to build and evaluate their designs.
The students will learn the importance of solution design through hands-on experience. This will help them understand the complexity of building real systems.

Assessment methods:

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students will learn how to design and implement various components of computer security. Since security solutions are threat specific, they will understand the overall engineering design process (identification of constraints, description of design criteria and objectives, usage of tools, development of a prototype, evaluation of the prototype based on the design criteria). Students will get hands-on experience in proposing, designing and executing the project.
Students will also learn how to work in teams; each student will actively participate as a member of a team, collaborate with the other team members, share the daily design activities and management of the project and contribute to achieve the project goals. At the end of the project, they will have to write a team report that describes their design, thus they will learn how to communicate their ideas effectively.

Relationship of course to program outcomes: The contribution of CS165 to program outcomes (a)-(k) or (1) – (9) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand basic issues to be addressed secure computer systems design, including risk, privacy, integrity, and availability.	2	2	0	0	0	2	0	0	0	0	3
Understand the basic security and threat models.	3	0	0	0	0	1	0	0	0	1	3
Understand the principles of cryptography.	2	0	0	0	0	1	0	0	0	0	3
Understand various protocols for authentication, data security and privacy, key generation and management, and application-specific security.	2	0	0	0	0	1	0	0	0	1	3
Understand program security, including malicious code and countermeasures.	1	0	0	0	0	2	0	0	0	0	3
Understand the principles of secure systems design.	1	0	1	0	0	1	0	0	0	0	3

Understand the principles of network and distributed systems security.	1	0	0	0	0	1	0	0	0	0	3
Develop an appreciation of issues systems administration.	1	0	0	0	0	1	0	0	0	0	3
Write programs to reinforce concepts learned.	3	1	2	0	0	0	0	0	0	0	3

Prepared by, and date of preparation: Srikanth Krishnamurthy, June 15, 2006

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 166	Course Title	Database Management Systems
Units	4	Course Coordinator	Vassilis Tsotras
Required/elective	Elective	URL (if any):	http://www.cs.ucr.edu/cs166/

Current Catalog Description:

Topics include architecture of database management systems; relational, network, and hierarchical models; distributed database concepts; query languages; implementation issues; and privacy and security of the database.

Textbook

Database Management Systems by Raghu Ramakrishnan and Johannes Gehrke, McGraw-Hill, Third Edition

References/Materials

http://www.cs.ucr.edu/cs166/main_manual.htm - Short Postgres Database tutorial
http://developer.postgresql.org/docs/postgres/ - Extensive Postgres Database development documentation
http://www.cs.ucr.edu/cs166/project.htm - Supporting materials for the course project

Course Goals/Objectives

- Understanding the basic architecture of relational Database management systems .
- Model real-life applications using the elements of Entity Relationship model: entities, attributes, relations, participation constraints, aggregations, hierarchy and how to convert this ER model in efficient Database schema.
- Learn the theoretical models of relational query languages described in terms of Relational algebra and relational calculus
- Understand how to manipulate data and write complex queries in the relational database model using SQL.
- Capture the fundamentals of physical database organization.
- Learn different techniques for data indexing and data organization, and their applications to improve the database retrieval performance.
- Learn the basic algorithms for efficient data access and how to estimate their cost in terms of I/O operations.
- Understand Query Optimization.

Prerequisites by Courses and Topics

CS 141. Intermediate Data Structures and Algorithms

Major Topics Covered in the Course

Steps in the database design process: Conceptual design and Logical database design, Entity relationship models. Fundamentals of the query languages. Relational algebra operators: selection, projection, Cartesian product, solving relational algebra problems. Relational calculus and relational-complete languages. Data manipulation using SQL. Writing database queries in SQL: single block and correlated queries.
Fundamentals of the database architecture and organization. Indexing: B+ tree, Hash based indexing. Algorithms for data processing: external sort algorithm, Join algorithms. Relational Query Optimization.

Laboratory schedule: number of sessions per week and duration of each session

Lecture, 3 hours; Lab section, 3 hours.

Lab 1 - Introduction to Database Management Systems
Lab 2 - Introduction to Database Design – ER modeling
Lab 3 - Relational model
Lab 4 - Relational algebra
Lab 5 - Structured Query Language - SQL
Lab 6 - Structured Query Language - SQL (part 2)
Lab 7 - Working with JDBC
Lab 8 - Working on the course project
Lab 9 - Working on the course project
Lab 10 - Working on the course project

Laboratory projects (specify number of weeks on each)

Implementation of information system using relational database for storage
Phase 1 – ER design: 3 weeks
Phase 2 – Relational schema design: 1 week
Phase 3 – Implementation (SQL queries, client application development): 6 weeks

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	5	Data Structures	15	5
Software Design	20	15	Prog. Languages	10	10
Comp. Arch.	5	5			

Oral and Written Communications:
Every student is required to submit at least __2___ written reports (not including exams, tests, quizzes, or commented programs) of typically __4-5__ pages and to make _ 1 __ oral presentation of typically _ 20 __ minutes duration.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

N/A

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Introduction & Relational Model – 3 lectures
Relational Algebra and Relational Calculus – 2 lectures
SQL - 3 lectures
Storage and Indexing – 1 lecture
Storing Data: Disks and Files – 1 lecture
Tree-Structured Indexing (B+ Tree) – 1 lecture
Hash-Based Indexing – 2 lectures
Query Evaluation – 2 lectures
External Sorting – 1 lecture
Evaluation of Relational Operators – 2 lectures
Relational Query Optimizer – 1 lecture

Problem Analysis

Please describe the analysis experiences common to all course sections.

During the course the students have to learn how to analyze the project requirements. The outcome of this process is project specification based on the requirements. Instructions are given in the first two lectures covering the Design steps and ER model and also in the first lab section.

Solution Design

Please describe the design experiences common to all course sections.

Based on the project specification, the students have to show their designing skills creating Entity relationship diagram for their project. Using this diagram they have to create Relational database design which later they have to implement. During the third phase of the project the students have to design and implement client side application for their projects.

Assessment methods

Midterm Exam: ~ 25%
Final Exam: ~ 35%
Homework Assignments: ~ 10%
Project: ~ 25%
Lab attendance: ~ 5%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students learn how to design a real life database application from scratch. Every year the project covers a different example (recent example projects are: design a hotel reservation system, a car rental application, etc.) They then use a relational database management system to implement the application, including simple front-ends. They work in groups so they have to learn how to cooperate, manage their time and complete the project. They also have to write a report and present their project which prepares them for realistic work environments.

Relationship of course to program outcomes: The contribution of CS166 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand advantages of using a Database Management System as a tool to store maintain and query large amounts of data.	1	0	0	0	0	0	0	0	0	0	3
Understand the principles of Database Design through the Entity-Relationship approach.	2	0	0	0	0	0	0	0	0	0	3
Understand the foundations of the Relational Model.	3	0	0	0	0	0	0	0	0	0	3
Understand the meaning of relationally complete systems with emphasis on Relational Algebra as a basic query language.	3	0	0	0	0	0	0	0	0	0	3
Learn the basics of SQL the standard commercial relational language.	2	0	0	0	0	0	0	0	0	0	3
Learn the basics of SQL the standard commercial relational language.	1	0	0	0	0	0	0	0	0	0	3
Learn the principles of access methods with emphasis on B+-treesand external hashing.	2	0	0	0	0	0	0	0	0	0	3
Understand Query Optimization.	3	0	0	0	0	0	0	0	0	0	3
Understand the basics of transaction processing.	2	0	0	0	0	0	0	0	0	0	3
Develop a complex application from design to query/form implementation using a relational database management system.	3	0	3	0	2	0	0	0	0	0	3

Prepared by, and date of preparation:
Vassilis Tsotras – 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 168	Course Title	Introduction to Very Large Scale Integration (VLSI) Design
Units	5	Course Coordinator	Harry Hsieh / Frank Vahid / Roman Lysecky
Required/elective	elective	URL (if any):	http://www.cs.ucr.edu/cs168/cs168-04win

Current Catalog Description: Basic electrical properties of metal-oxide-semi conductor (MOS) circuits. MOS circuit design processes. Basic circuit concepts. Subsystem design and layout. Aspects of system design. Memory, registers, and aspects of systems timing.

Textbook:

References/Materials

Modern VLSI Design: System-on-Chip Design by *Wayne Wolf, Third Edition, Prentice Hall PTR*

Course Goals/Objectives:

- Understand basic operation of CMOS transistors
- Be able to build simple logic gates from transistors
- Be able to layout simple logic gates
- Be able to create layouts for basic combinational and sequential components
- Understand modern chip design and fabrication issues, challenges and trends
- Understand how circuits are mapped to FPGAs

Prerequisites by Courses and Topics: CS 120A/EE 120A: Logic Design. The design of digital systems. Topics include Boolean algebra; combinational and sequential logic design; design and use of arithmetic-logic units, carry-look-ahead adders, multiplexors, decoders, comparators, multipliers, flip-flops, registers, and simple memories; state-machine design; and basic register-transfer level design; or consent of instructor.

Major Topics Covered in the Course

Lecture Topic includes: Introduction to VLSI Design; Fabrication, Transistor Structures, Basic Transistor Behavior; Transistor Characteristics; Wires, Vias, and Par1asitics; Design Rules and Stick Diagrams; Combinational Logic Functions and CMOS Logic Gates; Electrical Properties of Combinational Gates; Wire Delay, Buffer Insertion; Pseudo nMOS Gates, DCVS Logic, Domino Gates; Layout, Channel Routing, Simulation; Combinational Network Delay, Logic Optimization; Transistor Sizing; Interconnect Design, Crosstalk, Power Optimization; Switch Networks, Combinational Testing; Memory Elements, Basics of Sequential Machines; Clocking Disciplines; Sequential Machine Design; State Assignment, Power Optimization, Design Validation, Sequential Testing; FPGA Fabric Architecture; SRAM-based FPGA Fabrics; Shifters, Adders, ALU; Multipliers; Memories, Datapaths, PLAs.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 6 hours.

Laboratory projects (specify number of weeks on each)

52. Introduction to Cadence Schematic Design/ Simulation (0.5 weeks)
53. Introduction to Cadence Layout Design (0.5 weeks)
54. NAND Gate Transistor/Layout Design (1 week)
55. 4-bit Adder Design (1.5 weeks)
56. 1-bit SRAM Memory Cell Design (1 week)
57. 4-bit SRAM Shift Register (1.5 weeks)
58. FPGA CLB Design (3 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		20%	Data Structures		10%
Software Design	5%	15%	Prog. Languages	5%	10%
Comp. Arch.	10%	25%			

Oral and Written Communications:
Every student is required to submit at least __7___ written reports (not including exams, tests, quizzes, or commented programs) of typically ___3__ pages and to make __4___ oral presentations of typically __5___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

First week discusses (in lecture) subject of applying VLSI design/systems for improved quality of life and safety, including discussion of systems like medical devices, automobiles, robots, etc. Student presentations include application of embedded system to socially related projects (e.g. earthquake alarm, alternative energy sources, medical robots).

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

- Principle of Digital systems and VLSI (1 lecture hour)
- Transistor and Layout Structures (4 lecture hours)
- Designing with Logic Gates (5 lecture hours)
- Building Combinational Logic Networks (5 lecture hours)
- Principle of Sequential Machines (4 lecture hours)
- Subsystem Design (3 lecture hours)
- FPGA Archiecture (2 lecture hours)

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analysis of area/performance/power/design-time/reliability of advance VLSI sdesign which may consists of both different structure, different design method and algorithms. Extensive trade-off analysis throughout.

Solution Design

Please describe the design experiences common to all course sections.

Lecture: Design of advance VLSI systems building up from Transistor layout, to logic gates, to combinational logic networks, to sequential machines, to the entire sussystem.
Lab: Design of advance VLSI systems building up from gate, to adder, to memory cell, to registers, and to the entire FPGA CLB..

Assessment methods:

Lecture: Midterm exam, final exam, homeworks.

Lab: Demo, reports, code, and exam.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Emphasizes tradeoffs among different implementation options, especially tradeoffs among use of different circuit structures, different algorithms, different techniques for low power and reliability, and different tools to achieve the metrics. The metrics for trade-offs are performance/power/area/design_time/reliability.

Relationship of course to program outcomes: The contribution of CS168 to program outcomes (a)-(k) or (1) – (10) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Understand basic operation of CMOS transistors	3	0	0	0	0	0	0	0	0	0	3
Be able to build simple logic gates from transistors	3	0	1	0	0	0	0	0	0	0	3
Be able to layout simple logic gates	3	0	2	0	0	0	0	0	0	0	3
Be able to create layouts for basic combinational and sequential components	3	0	2	0	1	0	0	0	0	0	3
Understand modern chip design and fabrication issues, challenges and trends	1	0	0	0	0	0	0	0	0	2	3
Understand how circuits are mapped to FPGAs	1	0	0	0	0	0	0	0	0	0	3

Prepared by, and date of preparation: Tom Payne, June 15, 2006

COURSE DESCRIPTION

Dept., Number	CS, 170	Course Title	
Units		Course Coordinator	Christian Shelton
Required/elective		URL (if any):	

Current Catalog Description

Introduction to fundamental problems underlying the design of intelligent systems and to one of the languages of artificial intelligence such as Prolog or LISP. Topics include brute force and heuristic search, problem solving, knowledge representation, predicate logic and logical interference, frames, semantic nets, natural language processing, and expert systems.

Textbook

Artificial Intelligence: A Modern Approach, second edition by Russell & Norvig

References/Materials

none

Course Goals/Objectives

1:Learn the basic principles and techniques that have been developed to address artificial intelligence, the problems for which they are applicable, and their limitations.
2: Learn how to represent problems as search problems (defining ``states", ``operators", ``search spaces" etc).
3:Learn when/how to use blind search techniques.
4:Learn when/how to use heuristic search, how to design problem specific heuristics.
5:Learn when/how to use optimizing search, how to quantify the trade-offs of time/space complexity and solution quality.
6:Learn when/how to use adversarial search (Game playing search). How to design evaluation functions. How to deal with probabilistic games/incomplete information games.
7: Learn how to represent knowledge in various logical representations, including propositional logic and first order logic, using syntax and semantics.
8: Learn how to manipulate facts in logic representations. Theorem proving (resolution) in logic systems.
9: Learn how to prepare/preprocess data to enable classification (for machine learning), including discretization, dimensionality reduction, feature generation etc.
10: Learn the advantages and disadvantages of the major classification algorithms (decision trees, Bayes classifier etc). Which types of problems each may be suitable for.

Prerequisites by Courses and Topics

CS 141

Major Topics Covered in the Course

AI methods: blind search, optimizing search, A* search, heuristics, predicate logic, propositional logic, entailment, resolution, probabilities, machine learning, classification, decision trees

Laboratory schedule: number of sessions per week and duration of each session

1 3-hour session per week

Laboratory projects (specify number of weeks on each)

Programming problems in homework: 3, each for 2 weeks.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	15%	10%	Data Structures	50%	
Software Design			Prog. Languages		5%
Comp. Arch.					

Oral and Written Communications:

Every student is required to submit at least _0___ written reports (not including exams, tests, quizzes, or commented programs) of typically _____ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Turing test and quantification of intelligence covered including methods and current state-of-the-art. Discussion of current AI systems is included (autonomous driving, care for elderly, for example) and tangentially their impact on society. Not graded.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Propositional logic and predicate logic are covered as well as their relationship to programming (through Prolog). Graph search and heuristics as lower bounds are covered. Classification as decision boundaries in high-dimensional spaces is covered.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Running time of graph search algorithms, analysis of heuristic's validity (consistency and admissibility), introduction to learning complexity (analysis of classifier's complexity).

Solution Design

Please describe the design experiences common to all course sections.

Students must match problem specification to abstract solution concept (search, logical entailment, or classification) and demonstrate how the solution method solves the problem.

Assessment methods

Four problem sets (50% total), one midterm (20%), one final (30%)

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students are exposed to and write programs with common AI algorithms (search, entailment, classification) and see the limits and powers associated with each. Emphasis is placed on students understanding of which solutions are applicable to which problem specifications.

Relationship of course to program outcomes: The contribution of CS170 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Learn the basic principles and techniques that have been developed to address artificial intelligence the problems for which they are applicable and their limitations.	2	0	0	0	0	0	0	0	0	0	3
Learn how to represent problems as search problems (defining ``states" ``operators" ``search spaces" etc).	2	0	3	0	3	0	0	0	0	0	3
Learn when/how to use blind search techniques.	2	0	3	0	1	0	0	0	0	0	3
Learn when/how to use heuristic search how to design problem specificheuristics.	1	0	3	0	1	0	0	0	0	0	3
Learn when/how to use optimizing search how to quantify the trade-offs of time/space complexity and solution quality.	3	0	3	0	3	0	0	0	0	0	3
: Learn when/how to use adversarial search (Game playing search). How to design evaluation functions. How to deal with probabilistic games/incomplete information games.	3	0	3	0	3	1	0	0	0	0	3
Learn how to represent knowledge in various logical representations including propositional logic and first order logic using syntax and semantics.	3	0	0	0	0	0	0	0	0	0	3
Learn how to manipulate facts in logic representations. Theorem proving (resolution) in logic systems.	3	0	0	0	0	0	0	0	0	0	3
Learn how to prepare/preprocess data to enable classification(for machine learning) including discretization dimensionality reduction feature generation etc.	3	0	1	0	1	0	0	0	0	0	3
Learn the advantages and disadvantages of the major classification algorithms (decision trees Bayes classifier etc). Which types of problems each may be suitable for.	3	1	2	0	2	0	0	0	0	0	3

Prepared by, and date of preparation:
Teodor Przymusinski, 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 177	Course Title	Modeling and Simulation
Units	4	Course Coordinator	Mart Molle
Required/elective	elective	URL (if any):	http://www.cs.ucr.edu/~mart/CS_177_Home_Spring_2004.html

Current Catalog Description: Topics include validation of random number sequences; concepts in modeling and systems analysis; and conceptual models and their mathematical and computer realizations. Examines simulation modeling techniques including object-oriented modeling and discrete-event modeling. Emphasis is on the use of simulation libraries used with programming languages such as C++. Requires a term project consisting of the development, computer implementation, and analysis of a model.

Textbook: A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis* (third edition), McGraw-Hill 2000.

References/Materials

- CSIM-19 Reference Guide, Mesquite Software, 2005 http://www.mesquite.com/documentation/index.htm
- M. Molle, Supplementary Lecture Notes on Discrete Event Simulation, http://www.cs.ucr.edu/~mart/177/simulation_notes.pdf

Course Goals/Objectives:

1. Learn the process for developing functional models that can be used to study the behavior of a dynamic system
2. Learn to apply elementary results from probability and statistics to modeling problems, including the values of random variables and the times of random events
3. Learn how to analyze the output from a set of simulation experiments that include random factors, to determine the statistical significance of those results.
4. Learn how experimental design can be used to increase the amount of information you obtain from a given set of experiments
5. Learn to write programs using both the event-driven and process-interaction paradigms

Prerequisites by Courses and Topics: CS 141: Intermediate Data Structures and Algorithms. Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming.

Major Topics Covered in the Course

- Week1: Introduction to Simulation and Modeling
- Week2: Constructing a Simulation Model in a General Purpose Programming Language
- Week3: Concurrent Process Interaction models using the CSIM-19 Package
- Week4: Review of Probability and Statistics
- Week5: Confidence Intervals and Hypothesis Testing
- Week6: Selecting Input Distributions
- Week7: Random Number Generation
- Week8: Other Simulation Languages
- Week9: Output Analysis and Run Length Control
- Week10: Variance Reduction Techniques

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

- Lab 1: (1 week) Study the program structure for a moderately-complicated simulation program written in a general purpose language, using the gas station example described in the supplementary notes. Compile and run the program after making a few minor changes.
- Lab 2: (1 week) Learn to use the CSIM-19 package, and rewrite the gas station example as a CSIM model.
- Lab 3: (2 weeks) Design and implement a complete simulation model for a simple system in a general purpose language.
- Lab 4: (1 week) Use the built-in random number generators in CSIM to experimentally validate the tabulated values for the standard normal and t-distributions.
- Lab 5: (2 weeks) Design and implement a complete simulation model for a more complex system using CSIM-19.
- Lab 6: (1 week) Collect measurement data for the final experiment and pool the results with other students to create a large empirical data set.
- Lab 7: (2 weeks) Incorporate the empirical data collected in Lab 6 into the complex CSIM-19 simulation model developed in Lab 5 to carry out an extensive measurement study, including proper output analysis.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	10	10	Data Structures	5	5
Software Design	5	5	Prog. Languages	5	5
Comp. Arch.	5				

Oral and Written Communications:
Every student is required to submit at least ___1__ written reports (not including exams, tests, quizzes, or commented programs) of typically __6___ pages and to make _____ oral presentations of typically _____ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Introduction to simulation and modeling describes applications of this methodology to topics with significant social and ethical relevance, such as environmental models, risk assessment in systems subject to catastrophic failure (eg., dams to survive once-per-century flooding), and training systems for dangerous activities. It is covered in short-answer test questions. (1 lecture)

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Efficient data structures to support priority queues, which represent the continuously-updated list of pending events, such as lists, heaps, and calendar queues (1 hour). Core topics from probability and statistics, including the Central Limit Theorem; Strong Law of Large Numbers; confidence intervals; hypothesis testing; independence; random processes (including transient versus equilibrium behavior); calculation of the moments of a probability distribution (including lost precision due to numerical cancellation errors); random number generators (including tests for uniformity and independence); tests for goodness-of-fit (8 hours).

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions. More specifically, students will apply rigorous statistical analysis to determine the significance of their experimental results via confidence intervals, variance reduction techniques, etc. Students will gather empirical data to parameterize their models and compare their data to theoretical distributions.

Solution Design

Please describe the design experiences common to all course sections.

Understanding the process of model formulation: separating the system from its environment; identifying the significant components of its state and possible events leaving to state changes; validation. Learning to program in a multi-threaded environment, including race conditions, critical sections, signals, queues and other inter-process communication paradigms. Evaluating the success of their design by testing against known benchmarks, the use of assertions to catch anomalous behavior.

Assessment methods:

20% Programming problems
20% Final Project
20% Midterm test
40% Final exam

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students must gather their own empirical measurement data to parameterize the experiments in the Final Project. Students must report the level of significance of their findings, i.e., not only that customer waiting times were reduced by an average of 5 minutes under the improved scheduling plan, but that the 95% confidence interval for the magnitude of the improvement was +/- 6 minutes, and hence explain that their results are inconclusive without additional tests.

Relationship of course to program outcomes: The contribution of CS177 to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Learn process for creating functional models	3	0	1	0	0	0	0	0	0	0	3
Learn to apply elementary probability and statistics, including values for random variables and times for random events	3	3	0	0	0	0	0	0	0	0	3
Learn output analysis from a set of experiments to determine its statistical significance	3	3	0	0	0	0	0	0	0	0	3
Learn to use experimental design to increase the information provided by a given set of experiments	3	3	0	0	0	0	0	0	0	0	3
Learn to write programs using both the event-driven and process-interaction paradigms	3	0	2	0	2	0	0	0	0	0	3

Prepared by, and date of preparation:

Mart Molle, June 20, 2006

COURSE DESCRIPTION

Dept., Number	CS 179	Course Title	Project in Computer Science
Units	4	Course Coordinator	Christian Shelton
Required/elective	required	URL (if any):	

Current Catalog Description

Under the direction of a faculty member, students (individually or in small teams with shared responsibilities) propose, design, build, test, and document software and/or hardware devices or systems. Requires a written report, giving details of the project and test results, and an oral presentation of the design aspects. Emphasizes teamwork, making technical presentations, and developing oral and written communication skills.

Textbook

varies from offering to offering

References/Materials

ACM Code of Ethics and Professional Responsibility, other topical information, plus additional material that varies from offering to offering.

Course Goals/Objectives

1: balancing design tradeoffs: cost performance schedule and risk
2: Writing project proposals
3: Team-project organization and management (including time lines)
4: Requirements capture and analysis
5: design and architecture
6: prototyping (possibly via simulation)
7: verification/validation
8: writing and presenting final reports
9: engineering professionalism and responsibility
10: engineering careers and the modern world

Prerequisites by Courses and Topics

CS 141 with a grade of "C-" or better; ENGR 180; 12 additional upper-division units in Computer Science. Each offering of CS179 is oriented toward a particular topic such as compilers, in which case it would have our upper-division course on compilers, CS152, as a prerequisite. Similarly for all other topics.

Major Topics Covered in the Course

teamwork, technical presentations, oral and written communication, various technical topics as per course offering

Laboratory schedule: number of sessions per week and duration of each session

nine lab-hours per week, three of which are rigidly scheduled per week, the remainder as desired by the students

Laboratory projects (specify number of weeks on each)

project proposal (2 weeks), design specification (2 weeks), final project (10 weeks)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		10%	Data Structures		
Software Design		15%	Prog. Languages		
Comp. Arch.					

Oral and Written Communications:

Every student is required to submit at least __1___ written reports (not including exams, tests, quizzes, or commented programs) of typically __20___ pages and to make __1___ oral presentations of typically __15___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Professional codes of ethics, social implications of created artifacts including current modern systems. Discussion of current systems in the topic area. Final report must include section on societal impact.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Varies by topic. Usually advanced algorithms or architectures required for design project.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students must take a user-specified task and break it into engineering components.

Solution Design

Please describe the design experiences common to all course sections.

Students must take the components and design, specify, and build each one to complete the final project.

Assessment methods

Preliminary Specifications (20%), Teamwork/weekly reports (20%), Final Project (20%), Final Report (20%), Final Presentation (20%)

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

This course directly involves the student in the identification of an engineering problem, the design and specification of a system to meet the problem's needs, the implementation of the solution, and the presentation of the results. Throughout the course, the students work in teams and must deal with team dynamics and scheduling. Students are responsible for making periodic "reports" to the project manager (the instructor).

Relationship of course to program outcomes: The contribution of CS179 to program outcomes (a)-(k) or (1) – (13) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Balancing design tradeoffs: cost performance schedule and risk	2	0	3	3	3	0	0	0	0	0	3
Writing project proposals	0	0	0	3	3	0	3	0	0	0	3
Team-project organization and management (including time lines)	0	0	2	3	0	0	3	0	0	0	3
Requirements capture and analysis	2	0	3	3	3	0	0	0	0	0	3
Design and architecture	2	0	0	3	3	0	0	0	0	0	3
Prototyping (possibly via simulation)	3	3	0	3	3	0	0	0	0	0	3
verification/validation	3	3	0	3	0	0	0	0	0	0	3
8: writing and presenting final reports	0	0	0	3	0	0	3	0	0	0	3
engineering professionalism and responsibility	0	0	0	0	0	3	0	0	0	1	3
engineering careers and the modern world	0	0	0	0	0	3	0	3	3	3	3

Prepared by, and date of preparation:
Christian Shelton – 06/15/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 180	Course Title	Introduction to Software Engineering
Units	4	Course Coordinator	Teodor Przymusinski
Required/elective	elective	URL (if any):	

Current Catalog Description: A study of software engineering techniques for the development, maintenance, and evolution of large software systems. Topics include requirements and specification; system design and implementation; debugging, testing, and quality assurance; reengineering; project management; software process; tools; and environments.

Textbook:

References/Materials

- (Required) C. GHEZZI, M. JAZAYERI AND M. MANDRIOLI, Software Engineering, Prentice Hall, 2003, Sec. Edition, ISBN 0-13-305699-6
- (Recommended) John Sharp, Microsoft Visual C# 2005 Step by Step, ISBN 13-978-0-7356-2129-9, Microsoft 2005.

Course Goals/Objectives:

59. Provide students with a broad overview of software engineering, covering all phases of the software lifecycle and a variety of software process models
60. Provide students with a variety of techniques for requirements analysis, architectural and detailed design, validation and verification, as well as planning and management
61. Provide students with practical experience in applying such techniques by producing a (small) software product throughout the course and handing in certain documents as required in a ``classical" (i.e. non-agile) setting as milestones
62. Allow students to gain experience in scheduling and managing their projects using a hands-off approach to team formation, planning, and form of deliverables

Prerequisites by Courses and Topics: CS 141: Intermediate Data Structures and Algorithms. Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming.

Major Topics Covered in the Course

Introduction: Chapters 1-3.

Software Product: Chapters 4-6.

Software Process & Management: Chapter 7.

Selected case studies.

Selected topics from remaining chapters time permitting.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours Laboratory projects (specify number of weeks on each)

One week of orientation and general introduction to software engineering
One week of introduction to the language and the software environment (Visual Studio 2005)
Two weeks of preparation for the first team project
Two weeks of preparation for the second team project
One week of preparation for the midterm
Two weeks of preparation for the third team project
One week of preparation for the final

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Data Structures	5%	
Software Design	45%	20%	Prog. Languages	20%	10%
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least _6____ written reports (not including exams, tests, quizzes, or commented programs) of typically __2-3___ pages and to make __0___ oral presentations of typically __0___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues
Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

The are taught the golden rule with respect to error messages, and are shown easy ways to make error messages more helpful. The only feedback on that is in the form of advice from the TAs during labs.

The students are lectured on the importance of high-level programming languages to programmer productivity and the importance of programmer productivity to all of information technology.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

The theoretical content is limited due to the fact that the course mostly concentrates on the principles and practice of software design

Problem Analysis

Please describe the analysis experiences common to all course sections.

All teams must prepare software specification and software design documentation for their projects and follow it up with the careful analysis of project design and implementation at its completion

Solution Design

Please describe the design experiences common to all course sections.

Students learn how to collaborate with one another by working in teams, to follow principles of structured design and proper software testing and assessment methods. We also emphasize proper documentation techniques

Assessment methods:

Quizzes and Lab assignments 10%

Midterm and Final Exam 50%

Programming assignments 40%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Software design process is of course a special case of the engineering software design process. Thus, in essence, the entire course is devoted to preparing students for engineering practice by learning to apply proper software, design, testing and evaluation principles and techniques

Relationship of course to program outcomes: The contribution of CS 180 to program outcomes (a)-(k) or (1) – (4) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially

Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Provide students with a broad overview of software engineering, covering all phases of the software lifecycle and a variety of software process models	0	0	0	0	0	1	0	0	0	0	3
Provide students with a variety of techniques for requirements analysis, architectural and detailed design, validation and verification, as well as planning and management	3	2	1	0	0	0	0	0	0	0	3
Provide students with practical experience in applying such techniques by producing a (small) software product throughout the course and handing in certain documents as required in a ``classical" (i.e. non-agile) setting as milestones	1	1	3	0	2	2	2	0	0	0	3
Allow students to gain experience in scheduling and managing their projects using a hands-off approach to team formation, planning, and form of deliverables	1	1	2	0	2	2	2	0	0	0	3

Prepared by, and date of preparation:
Teodor Przymusinski 6/22/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 181	Course Title	Principles of Programming Languages
Units	4	Course Coordinator	Teodor Przymusinski
Required/elective	elective	URL (if any):	

Current Catalog Description: Principles of programming language design. Study and comparison of several programming languages, their features and their implementations.

Textbook:

References/Materials

C. Ghezzi and M.Jazayeri, Programming Language Concepts, John Wiley.
Textbook's home page: http://www.infosys.tuwien.ac.at/pl-book/

Course Goals/Objectives:

63. Provide students with a broad background in programming languages, working from principles and theoretical foundations as well as concrete example languages
64. Theoretical foundations include formal syntax, lambda calculus, formal semantics, type systems and inference, etc.
65. Concrete example languages include Scheme, Prolog, ML, Modula-3, Simula, Smalltalk, C++, and Java
66. Provide students practical experience in applying various languages to (really small) problems
67. Encourage students to study on their own either (1) a separate language, writing a tutorial on it or (2) investigate one or more language constructs in detail by developing a suitable interpreter

Prerequisites by Courses and Topics: CS 061: Machine Organization and Assembly Language Programming; CS 141: Intermediate Data Structures and Algorithms (CS 141 may be taken concurrently); CS 150: Theory of Automata and Formal Languages.

Major Topics Covered in the Course

- Chapter 1: Introduction.
- Chapter 2: Syntax and Semantics.
- Chapter 3: Structuring the Data
- Selected sections from Chapters 4, 5, 6, 7, 8
- Brief introduction to Prolog and Java.

Laboratory schedule: number of sessions per week and duration of each session:
Lecture, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

One week of orientation and general introduction to programming languages
Two weeks of preparation for the first project
Two weeks spent on learning the SIMPLESEM
One week of preparation for the midterm
Two weeks of preparation for the second project
Two weeks of brief intro to two non-procedural languages, e.g., Prolog and LISP
One week of preparation for the final

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5%		Data Structures	5%	
Software Design			Prog. Languages	70%	20%
Comp. Arch.					

Oral and Written Communications:
Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) of typically ___0__ pages and to make __0___ oral presentations of typically __0___ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

The are taught the golden rule with respect to error messages, and are shown easy ways to make error messages more helpful. The only feedback on that is in the form of advice from the TAs during labs.

The students are lectured on the importance of high-level programming languages to programmer productivity and the importance of programmer productivity to all of information technology.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Students learn basic principles of programming language design. Theoretical foundations include formal syntax, lambda calculus, formal semantics, type systems and inference, etc.

Problem Analysis

Please describe the analysis experiences common to all course sections.

Analytical experience is limited to the testing and evaluation of student projects

Solution Design

Please describe the design experiences common to all course sections.

Design experience is limited to the design of student projects

Assessment methods:

Midterm Exam 20%

Final Exam 45%

Programming assignments (2) 25%

Lab work 10%

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

By learning principles of programming languages students will not only better understand the features and the limitations of languages they are familiar with but will be much better prepared to learn new languages which will inevitably appear in the new future.

Relationship of course to program outcomes: The contribution of CS 181 to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
Provide students with a broad background in programming languages, working from principles and theoretical foundations as well as concrete example languages	2	0	0	0	0	0	0	0	0	0	3
Theoretical foundations include formal syntax, lambda calculus, formal semantics, type systems and inference, etc.	3	0	0	0	0	0	0	0	0	0	3
Concrete example languages include Scheme, Prolog, ML, Modula-3, Simula, Smalltalk, C++, and Java	2	0	0	0	0	0	0	0	0	0	3
Provide students practical experience in applying various languages to (really small) problems	2	0	1	0	1	0	0	0	0	0	3
Encourage students to study on their own either (1) a separate language, writing a tutorial on it or (2) investigate one or more language constructs in detail by developing a suitable interpreter	1	0	0	0	0	1	1	0	3	1	3

Prepared by, and date of preparation:
Teodor Przymusinski 6/22/06

COURSE DESCRIPTION

Dept., Number	Computer Science and Engineering, CS 183	Course Title	UNIX System Administration
Units	4	Course Coordinator	Victor Hill
Required/elective	elective	URL (if any):	

Current Catalog Description: Technical aspects of system administration on a Unix system including advanced Unix, managing system devices, operating system installation, communications, and networking.

Textbook: *UNIX System Administration Handbook*, 3rd Edition, Evi Nemeth, Garth Snyde, Scott Seebass, Trent Hein

References/Materials

http://moodle.cs.ucr.edu/moodle/course/category.php?id=10
Course webpages for offerings of CS 183

Course Goals/Objectives:

- Demonstrate proficiency in installing, using, and administrating UNIX-based operating systems.
- Create software tools to administrate systems tasks, including configuration and maintenance.
- Understand security issues including physical security, network security, and encryption.
- Project-based knowledge of fundamental network protocols.
- Installation and configuration of network services, including open-source file service, electronic mail, name service and other core network based services.
- Analysis and evaluation of best practices of system administration related to the topics above.

Prerequisites by Courses and Topics: CS 141: Intermediate Data Structures and Algorithms. Explores basic algorithm analysis using asymptotic notations, summation and recurrence relations, and algorithms and data structures for discrete structures including trees, strings, and graphs. Also covers general algorithm design techniques including "divide-and-conquer," the greedy method, and dynamic programming. Homework and programming assignments integrate knowledge of data structures, algorithms, and programming.

Major Topics Covered in the Course: UNIX environment: the command line, utilities, piping and redirection, interaction with the file system and processes. Operating System: installation and configuration of various distributions of Linux. User management: adding and removing accounts, and controlling resource access. TCP/IP networking: the

TCP/IP protocol stack, routing, security issues. Network Services: topics including domain name service, web, email, file, and authentication services.

Laboratory schedule: number of sessions per week and duration of each session:
Seminar, 3 hours; laboratory, 3 hours.

Laboratory projects (specify number of weeks on each)

Familiarity with UNIX environment (1)
Installation of Linux(1)
Installation and Configuration of Software (1)
Apache web server(1)
Postgresql database server(1)
LDAP and Kerberos directory services and authentication(1)
Samba file service(1)
Postfix mail server(1)
Antivirus/Antispam software(1)
Dovecot MAA server(1)

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	5	5	Data Structures	5	5
Software Design	15	25	Prog. Languages	10	20
Comp. Arch.	5	5			

Oral and Written Communications:
Every student is required to submit at least __2___ written reports (not including exams, tests, quizzes, or commented programs) of typically __5___ pages and to make __1___ oral presentations of typically ___15__ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Students learn the importance of social and ethical issues by working in teams and participating in discussions of network security and ethical responsibilities related to the system administration field.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

UNIX operating system – 6 lectures
TCP/IP networking – 6 lectures
Network Services – 3 lectures
Configuration management – 3 lectures

Problem Analysis

Please describe the analysis experiences common to all course sections.

Students will learn to analyze the requirements and goals of the projects, identify the important components and analyze the efficiency of their solutions.
The students will learn the importance of analysis through hands-on experience.

Solution Design

Please describe the design experiences common to all course sections.

A primary goal of the course is the design of solutions that meet project requirements. The students will design, implement and test software tools that automate system tasks and integrate related network services.
The students will learn the importance of solution design through hands-on experience. This will help them understand the complexity of building real systems.

Assessment methods: Homework / Labs, 30%; Projects, 25%; Exams, 45%.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students will learn how to design software that performs system maintenance and integrates the operation of numerous network services such as database, web, email, and authentication services, thus they will understand the overall engineering design process (identification of constraints, description of design criteria and objectives, usage of tools, development of a prototype, evaluation of the prototype based on the design criteria). Students will get hands-on experience in proposing, designing and executing the project.
Students will also learn how to work in teams; each student will actively participate as a member of a team, collaborate with the other team members, share the daily design activities and management of the project and contribute to achieve the project goals. At the end of the project, they will have to write a team report that describes their design, thus they will learn how to communicate their ideas effectively.

Relationship of course to program outcomes: The contribution of CS 183 to program outcomes (a)-(k) or (1) – (6) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K

Demonstrate proficiency in installing, using, and administrating UNIX-based operating systems.	1	0	0	0	0	0	0	0	0	0	3
Create software tools to administrate systems tasks, including configuration and maintenance.	1	0	2	0	2	0	0	0	0	0	3
Understand security issues including physical security, network security, and encryption.	1	0	0	0	0	2	0	0	0	0	3
Project-based knowledge of fundamental network protocols.	1	0	1	0	0	0	0	0	0	0	3
Installation and configuration of network services, including open-source file service, electronic mail, name service and other core network based services.	1	0	0	0	0	0	0	0	0	0	3
Analysis and evaluation of best practices of system administration related to the topics above.	0	0	0	0	0	1	0	0	1	1	3

Prepared by, and date of preparation: Victor Hill, June26, 2006

COURSE DESCRIPTION

Dept., Number	engr, 180	Course Title	Technical Communications
Units	4	Course Coordinator	S. Burton, B. Graham
Required/elective	required	URL (if any):	http://moodle.cs.ucr.edu

Current Catalog Description

Develops oral, written, and graphical-communication skills. Involves extensive oral communication and presentations in small groups, and preparing and critiquing reports, proposals, instructions, and business correspondence. Emphasizes professional and ethical responsibilities, and the need to stay current on technology and its global impact on economics, society, and the environment.

Textbook

The Inmates are Running the Asylum, Cooper
The Elements of Technical Writing

References/Materials

Course website.
http://www.vark.com
http://www.clearspecs.com

Course Goals/Objectives

1. an ability to participate and contribute to discussions and meetings, both in leading and nonleading roles.

2. an ability to make cogent, well-organized verbal presentations, with and without visual aids prepared via presentation software.

3. an ability to produce cogent, well-written documents (including email).

4. an understanding of professional and ethical responsibility, particularly regarding well-designed human interfaces including documentation.

5. an understanding of what is expected in the professional workplace, including the need for long-term professional development.

Prerequisites by Courses and Topics

ENGL 001C or ENGL 01SC

Major Topics Covered in the Course

Importance of communication in science and engineering, defining an audience, organizing and drafting documents, revising for organization and style, developing graphics, conducting meetings, memos/letters/email, proposals, progress reports, articles, instructions and procedures, electronic text, oral presentations, job search documents. Also, inductive and deductive reasoning, truth tables, presentation style and skills, VARK, use cases, mind maps, grammar and style, writing functional specifications, usability testing, explanations and simplification, drafting and revision, visual gestalt in design, designing for online use.

Laboratory schedule: number of sessions per week and duration of each session

One three-hour session per week

Laboratory projects (specify number of weeks on each)

Language arts diagnostics and resumes (1)
Diagramming and truth tables(1)
Writing for business final project lab work(1)
Persona and scenario writing (1)
Outlining a spec (1)
Functional spec draft(1)
Drafting and alpha doc test (1)
Revising and beta doc test (1)
Paper Airplane, revised(1)

Each session involves some of the following: oral presentations (with and without graphical aids), small group meetings (leading and non-leading roles), group problem solving, critiquing written and oral presentations.

Estimate Curriculum Category Content (percent of time)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	0	0	Data Structures	0	0
Software Design	0	0	Prog. Languages	0	0
Comp. Arch.	0	0			

Oral and Written Communications:

Every student is required to submit at least __15___ written reports (not including exams, tests, quizzes, or commented programs) of typically __2___ pages and to make ___1__ oral presentations of typically __15__ minute's duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

Social and Ethical Issues

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Ethical implications of poor documentation are discussed, and students are required to produce high-quality documentation and to rewrite poor documentation.

Theoretical Content

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Students are expected to understand critical thinking & logic as is applies to writing, and to synthesize that with other topics (~2 hours instruction, 3 hours lab).
Students are exposed to a variety of design & layout theories, including visual gestalt, and expected to discuss these topics not only theoretically, but articulate practical applications as well (~4-6 hours lecture, scattered throughout course)
Students are exposed to cognitive processing and learning theory, and how it applies to interfaces and documentation (4-6 hours lecture, scattered throughout course)

Problem Analysis

Please describe the analysis experiences common to all course sections.

Audience analysis – VARK, surveys, personas. Documentation analysis – fixing poor documentation, drafts, revisions, functional specification.

Solution Design

Please describe the design experiences common to all course sections.

Not applicable.

Assessment methods

40% written exams covering theory and writing, 10% homework,
30% presentations, 20% participation.

Students are checked off on work completed in labs, are graded on drafts, revisions, and completed documents, and take quizzes and exams that have multiple choice and essay questions.

Contribution of course to professional component: how the engineering experience gained here prepares student for engineering practice, e.g., how this engineering experience incorporates engineering standards and realistic constraints as described in EAC Criterion 4.

Students must learn to follow instructions, work in teams, give presentations, and produce documentation aimed at management, peers, and end users, to articulate technical information gracefully and usably, and to understand the key differences between themselves and their audience(s).

Relationship of course to program outcomes: The contribution of Engr 180 to program outcomes (a)-(k) or (1) – (5) is summarized in the objective-outcome matrix table.

Objective Outcome Matrix											
Objective Addresses Outcome: 1-slightly 2-moderately 3-substantially											
Outcome Related Learning Objectives	A	B	C	D	E	F	G	H	I	J	K
an ability to participate and contribute to discussions and meetings, both in leading and nonleading roles.	0	0	0	3	0	1	3	0	0	0	0
an ability to make cogent, well-organized verbal presentations.	0	0	0	3	0	0	3	0	0	0	0
an ability to produce cogent, well-written documents.	0	0	0	0	0	2	3	0	0	0	0
an understanding of professional and ethical responsibility, particularly regarding well-designed human interfaces including documentation.	0	0	0	0	0	3	0	0	0	0	0
an understanding of what is expected in the professional workplace, including the need for long-term professional development	0	0	0	0	0	3	0	0	0	0	0

A. Not applicable.
B. User survey and their analysis.
C. Functional specification.
D. The entire class, but most especially the material on user surveys and functional specifications.
E. The material functional specification and user analysis
F. The lectures *extensively* cover professional and ethical responsibility.
G. The entire class, e.g., the logic lectures strongly map to this, as does the presentation unit.
H. Same as G
I. The entire class.
J. Professional and ethical issues come up in the material on functional specification. "What do you do if you are asked to design something that is a "cheat" in that it short cuts and may be unsafe?"
K. The class as a whole.
Regarding assessment of the achievement of these outcomes, the deliverables at the end of each of the three units are the measurements. These are graded as though they are business deliverables in the workplace, because they are. The deliverables - resume, functional spec, presentation - are things they will deliver in the workplace one day. We

grade them as though these are from colleagues who want our opinion before they distribute.

The final exam contains questions targeting the outcomes that are covered. We especially look for synthesis of the concepts on the final. Those get full credit.

We can't measure whether or not they will do life long learning, but we sure talk about it a lot.

Prepared by, and date of preparation:
Tom Payne with help from Sharon Burton, Bonni Graham, and Victor Hill, 6/27/06