Computer Architecture
Syllabus of Qualifying Examination

PhD in Engineering with a focus in Computer Science

Reference course: CS 5200 Computer Architecture, College of EAS, UCCS

Created by Prof. Xiaobo Zhou, last updated February 2008

Description:

Computer architecture is the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance and cost goals. In this course, you will learn how to completely design a correct single processor computer, including processor datapath, processor control, pipelining optimization and instruction level parallelism, cache and memory systems, and I/O systems. You are going to see that no magic is required to design a computer. You will learn how to quantitatively measure and evaluate the performance of the designs.

This course serves you two ways. First, for those who will continue in computer architecture, it lays foundation of detailed experience necessary to take more advanced courses. Second, for those students not continuing in computer architecture, it gives an in-depth understanding of, and insights into, the inner-workings of modern computers, their evolutions, and trade-offs present at the hardware/software boundary. It also solidifies an intuition about why hardware is as it is.

Important issues of the course include, but not limited to, quantitative measure of the computer performance and design changes, design (and change) of single-cycle / multi-cycle datapath and control units according to specification of instruction sets, in-depth understanding of pipelining datapath and hazards, memory hierarchy organizations and caching schemes, etc.

Reference Books

A) J. L. Hennessey and D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufman, 4th edition (ISBN 0-12-370490-1)

B) D. A. Patterson and J. L. Hennessey, "Computer Organization and Design: The Hardware/Software Interface", Morgan Kaufman, third edition (ISBN 0-12-370606-5)

Coverage and Expectations

1. Knows how to quantitatively measure, summarize, and report computer performance. Important concepts include: CPI, IPC, instruction count, instruction mix, clock rate, MIPS rating, Amdahl's law, speed-up, benchmarking.

2. Have knowledge of the instruction set principles and examples. Important concepts include classification of instruction set architectures (ISA), Memory addressing, Type and size of operands, MIPS instruction set architectures

3. Learns to design and to change the designs according to a given ISA (e.g., MIPS). Important issues include design of single-cycle datapath and control units; design of multi-cycle datapath and control units; performance comparison of different designs

4. Understands the advanced pipelining design technologies. Important issues include design of the pipelining datapath based on MIPS ISA; Pipelining performance; Knowledge of pipelining hazards; Hardware forwarding and branch prediction; Basic compiler techniques for exposing Instruction level parallelism (ILP)

5. Understands the memory hierarchy designs and the concepts of locality, cache block, and address mapping. Important topics include cache organizations and associativity; measurement and evaluation of cache performance; reducing cache miss penalty, miss rate, hit time; virtual memory

6. Knows the basics of interfacing processors and peripherals, including storage and RAID systems, bus systems, and operating Systems' responsibilities in I/O operations

CS 5700 - Automata Theory, Languages and Computation - 3 Credit Hours

Textbook:

Introduction to Languages and the Theory of Computation, Third Edition, John C. Martin, McGraw-Hill, 2002.

Alternate References:

Introduction to Automata Theory, Languages and Computation, by Hopcroft & Ullman, Addison-Wesley, 2nd Edition, 2001.

Theory of Computation - Formal Languages, Automata and Complexity, by J. Glenn Brookshear, Benjamin-Cummings, 1989.

Textbook Coverage:

Chapters 1- 15 plus supplemental handouts (we will skip some parts of covered chapters)

Course Objective and Description:

The objective for this course is to introduce the fundamental concepts of computation theory as related to theoretical machines (automata, Turing machines), languages (with emphasis on formal languages and their grammars), and the concepts of computability and computational complexity. A major objective is to show the close relationships among theoretical machines, languages/grammars and computability/complexity. Emphasis for the course will be on theory coupled with practical applications. There are numerous homework assignments on the material.

Topical Outline:

	Topic	*Reference*
1.	Preliminaries: symbols, strings, alphabets and languages; graphs and trees, inductive proofs, sets and relations; Assignment: Prob set #1:	Ch 1-2
2.	Finite automata, regular languages, regular expressions, deterministic vs. non-deterministic FA and equivalency, regular grammars, variations on FA, Kleene's Theorem; Assignment: Read chapters 3-4, Prob sets #2 & #3	Ch 3-4
3.	Minimum-state FA, properties of regular sets, the pumping lemma, decision problems and algorithms; Assignment: Read chapter 5, Prob set #4	Ch 5
4.	Context-free grammars and languages, simplified forms (Chomsky and Griebach) of CFGs; Assignment: Read chapters 6; Prob set #5	Ch 6
5.	Pushdown automata, definition, relation to FA, relation to CFG/CFL, parsing; Assignment: read chapter 7, Prob set#6	Ch 7
6.	Properties of CFLs, pumping lemma, intersections and complements, decision problems; Assignment: Read chapter 8	Ch 8
7.	Turing machines, the TM model, computability & TMs, variations on TMs; Assignment: Read chapter 9, Prob set#7	Ch 9
8.	Recursively-enumerable languages, unsolvable decision problems, computability, primitive recursive functions, mu-recursive functions; Assignment: Read chapters 10-13	Ch 10-13
9.	Tractable and intractable problems, NP-complete problems; Assignment: Read chapters 14-15	Ch 14-15

Course Outcomes:

The following table lists expected outcomes on course completion. It defines knowledge a student should have.

Course Outcomes:

Understands grammars (G), languages (L) & automata (M)
Understands properties of sets, relations and mathematical induction
Understands the definition of grammars, languages and automata
Knows the differences between determinism and non-determinism
Knows the definition for regular grammars/languages & finite automata
Knows how to design a finite automaton to accept a regular language
Knows how to solve – given any one of (L, G, M), find the other two
Knows definition for context-free G & L, and pushdown automata
Knows how to design a pushdown automaton to accept context-free L
Knows definition for phrase-structured G & L and Turing machine
Knows how to design a Turing machine to accept any language
Understands difference between acceptability and decideability
Understands definitions for computational complexity
Understands the concept of computability
Understands the hierarchical relationships for languages and automata

CS 5500 - Operating Systems I

Textbook: "Modern Operating Systems" by Andrew S. Tanenbaum (2nd Edition)

Topics:

I. Introduction and historical perspective
1. Terminology
2. Purposes and Techniques
3. Historical Perspective

II. Input/Output
1. Synchronous I/O
2. Asynchronous I/O
3. I/O buffering
4. I/O management

III. Cooperating Processes
1. Definitions
2. Critical Sections
3. Synchronization and Communication
4. Primitive Operations and their implementation

IV. Multiprogramming Systems
1. Basic Hardware/Software Needs
2. Protection and Security
3. Systematic Design Methodologies
4. High and low level user interfaces

V. Resource Management
1. Reusable, consumable, and sharable resources
2. Static and dynamic management
3. Virtual mappings
4. Offline storage

VI. Process Management
   1. Primitive operations
   2. I/O processes
   3. Process Scheduling
   4. Multiple Processors

VII. Errors and Recovery
   1. Deadlock and algorithms for managing it.
   2. System Crashes

VIII. Distributed Systems
   1. Terminology
   2. Organizations
   3. Selected Topics

University of Colorado at Colorado Springs

CS 5720: Syllabus

Design and Analysis of Algorithms

Fall 2011

Text: *The Design and Analysis of Algorithms by Anany Levitin, 2nd Edition (*AL*)*

Mathematical Companion for Design and Analysis of Algorithms (Unfinished!) by Jugal Kalita (*JK*), available at http://www.cs.uccs.edu/~kalita/algorithms.pdf

Objective: Be familiar with algorithms used to perform a wide variety of tasks, and be able to carefully analyze them mathematically for their space and time complexities. Be able to program an algorithm in a language of choice to solve problems. Be able to perform experimental analysis of time and space requirements, and compare with theoretical analyses of the same.

Outline of Topics

Weeks	**Chapters**	**Topics**
1	AL 1	Introduction
2-4	AL 2, A, B; JK	Mathematical foundations: Focus on recurrence relations
5-6	AL 3	Brute Force Algorithms
7-8	AL 4	Divide-and-Conquer Algorithms
9-11	AL 8	Dynamic Programming
12-13	AL 9	Greeedy Techniques
14-15	AL 11, 12	NP-completeness and Approximation Algorithms