Computer Hardware Fundamentals
COSC-120, 2012, fall
Computer Science Department
Georgetown University

Richard K. Squier, Ph.D.
Phone: 202-687-6027
St. Mary's Hall, room 339
Office hours: 12:30-1:30 Tue/Thu, and by appointment

TA: TBA

Class Meetings

Tue/Thu 11:00-12:15, ICC 119

Course Description

After a brief historical overview, the course develops some theoretical ground work (Finite State Automata and Turing Machines) followed by some general design concepts (the von Neumann and Harvard models). The next section covers Boolean Algebra, logic gate implementations from mechanical switches to Very Large Scale Integration (VLSI), and an introduction to Digital Logic Design. The remainder of the course covers standard computer processor architecture and basic elements of computer system organization from a programmer's perspective using assembly language and C, based on the Patt & Patel text through chapter 10 and Appendices. In parallel with the above is a processor design project which implements Patt & Patel's LC-3 micro-architecture. The hierarchical implementation combines circuit design coupled with sub-modules implemented using the Verilog hardware description language. The circuits are laid out using Electric (an Electronic Design Automation tool) and the entire system is compiled to a complete Verilog description of the LC3, which is then simulated. LC3 assembly language programs are written, assembled, and hand linked, then executed by the simulation to exercise and test the LC3's various architectural features. Students manage their design source documents using Subversion, a source-code version-control system. The workload includes weekly assignments, mid-term exam, comprehensive final exam. Optional extra-credit projects could include a collaborative (but not team) semester CPU implementation project, a lab-bench electronic breadboard implementation of a small finite-state machine, or some other project. Prerequisites: COSC 052 (CS-2) and 030 (Math for CS) (previously numbered 072 and 127). Fall semester.

Course Goals

Our goal is to develop foundational concepts, principles, and analytic skill which can be generally applied in any computational setting, whether that be continued academic study, professional practice, or a personal context. We want to have the ability to intelligently approach any computational system or questions regarding their cost, use, practicality, computational power, future technological prospects, or theoretical interest; the ability to ask intelligent questions when we do not understand and appreciate the import of the answers, and the ability to educate ourselves when needed in the general area of computing systems. We want to have background knowledge of the concrete and detailed specifics of some example computing hardware that is generally representative. We want to acquire a level of skill in applying general principles to actual systems, recognizing technological tradeoffs involved, evaluating differing organizational implementations, and recognizing how instruction types and their execution requirements interact with the low-level hardware-software systems interface and thereby affect a system's applicability and performance. We want to understand how theoretical and implementation aspects of a system provide capabilities while simultaneously imposing limitations both on what we can do and how best to go about it. Finally, we want to develop our abstract and hierarchical systems thinking, our ability to think conceptually about parallel execution and programming, and acquaintance with tools and methods for system design, simulation, and verification.

Course Learning Objectives

To know
the basic difference between digital and analog computing;
the essential concepts of an effective procedure;
how to hierarchically compose functional elements;
what Turing universality has to do with programming language computational expressiveness;
the difference between source code and machine code;
how to use the Subversion commandline for file sharing and revision control;
the essential differences between a working copy and a repository copy;
how to design simple circuits and heirarchical systems in the Electric EDA;
the connection between Electric designs and the resulting extracted verilog simulation code;
how to write and use simple verilog testbenches;
the structure of the von Neumann architecture and its principle components in the LC3;
the phases of instruction execution and structure of instruction formats;
the nature of bits, bit order, bytes, byte order, words, and basic memory operations;
the meaning of machine state, state transition, input/output for finite state machines;
Mealy and Moore machines, state elements, next-state and output functions;
the Turing model of computation, finiteness, symbols and symbols sets, state and symbol encodings;
the congruences between Turing machines, programs, and hardware;
the difference between data-path state elements and control state elements;
the connections between register states, variable states, and machine states;
the congruence between computers and universal Turing machines, machine encodings and programs;
the simulation function of a universal Turing machine and composition of Turing machines;
the Halting Problem and the nonexistence proof by diagonalization and contradiction;
how to convert a verbal description of a system into a state-transition diagram description;
how to convert between a state-transition diagram and a Turing simulator's unary encoding;
how to convert a verbal description of a simple computational task to a Turing machine description;
the basic Boolean functions, composition, algebraic properties, min/max terms and expansions;
conversion from Boolean truth-table to AND-OR or OR-AND or NAND-NAND circuits;
the design and function of latches, flip-flops, and their use as FSM state or datapath elements;
elementary device physics and scaling effects on speed, design, clocking, heat, and market forces;
busses, transistors, tri-state buffers, basic CMOS logic gates, memory organization and interfaces;
logic design of decoders, muxes, demuxes, signal gating, PLAs,
basis of information and encoding, Shannon information theory, and error detection/correction;
complete languages, integer signed/unsigned arithmetic, 2's complement, floating point arithmetic;
LC3 datapath, ISA, components, control states and signals, memory-i/o busses and devices;
LC3 instruction execution, machine code, assembly programming, linking and assembly, debuggers;
RTL description, addressing modes and evaluation, memory delay, memory mapped i/o;
LC3 traps/exceptions/interrupts, stack operations, interrupt priority, privilege, polling;
basic use of unix shells, editors, make, regular expressions, sed, m4, tar, zip.

Required Text

Introduction to Computing Systems 2ed. Patt and Patel (McGraw-Hill, 2004, second printing)

Suggested Text

Computer Organization, Revised 4ed. Hennessy and Patterson (Morgan-Kaufmann, 2012)

Note that the Hennessy & Patterson (HP) text is used in the following course, cosc-121. It may be possible to use only this text, but there are many details in Patt & Patel that are not in HP.

Topic	**Patt & Patel**	**Patterson & Hennessey**
Introduction	Chp. 1	Preface; Chp. 1
Boolean Logic	Chp. 3	Appendix C
Logic Design	Chp. 3	Appendix C
Sequential Circuits	Chp. 3	Appendix C
Instruction Set	Chp. 5	Chp. 2,
Datapath	Chp. 4, 5	Chp. 4
Control	Chp. 4, Appendix C	Chp. 4, Appendix D
ALU, arithmetic, numbers	Chp. 2, 3	Chp. 3
Bit-level ISA	Chp. 6	Chp. 2
ISA and Compilers	Appendix A	Chp. 2, Appendix B
Assembly	Chp. 7	Chp. 2, Appendix B
I/O Basics	Chp. 8	Chp. 6
Traps, call-return	Chp. 9	Chp. 2, Appendix B
Interrupt I/O	Chp. 10	Appendix B
Performance	---	Chp. 1

Academic Integrity Policy

In assigning grades, one of my jobs as instructor is to ascertain your growth in understanding the intellectual content of the course during our studies together. Course assignments and projects are intended to facilitate that growth. However, at times, one's thinking can get lead astray by side-issues that may seriously hamper your efforts to understand. It is very important that you do not dwell fruitlessly on some point that has you stuck. You should seek help as soon as practical, and your classmates can be an efficient resource. For that reason, I encourage you to freely exchange information, and this Academic Integrity Policy is designed to allow for, and encourage that kind of cooperation. The default policy for the Computer Science Department is amended as follows. You are free to discuss problems and solutions of any assignment or project with your classmates or others. You need not cite these conversations nor indicate which parts of your submitted material was garnered from such conversations. You are free to collect information from any source, electronic or otherwise, and you need not indicate the original source nor that the material did not originate with you. In addition, in this context, I consider it a fault to withhold useful information from others; although, this policy makes no stricture against it. The ability to work cooperatively together is a learned skill that will be important later in life.

Grading

My grading system does not depend on evaluating your progress based on material of unknown origin. Homework is graded, but used solely to provide feedback, not in determining grades (However, see class participation below.) I do use your submitted material as a guide in developing examinations, and will ask that all your work be returned to me temporarily; so, keep ALL your work together as a portfolio. If you feel you are not being evaluated thoroughly enough, it is incumbent on you to bring this to my attention while there is still time to address your concerns before grades are submitted. You are welcome to discuss these issues with me at any time.

Midterm exam: 35%, Final Exam: 50%, Participation (in class discussions, and number and completeness of homework submissions) 15%. In addition, verifiable, documented, and significant extension of project work may receive up to a 100% boost in overall score. A project inquiry is either an individual oral question and answer session, or a written examination, or a class presentation of the project.

Homework markup system

A check mark means:
"I mostly agree with what you said", "I cannot find anything worth quibbling about", "Technically the answer is correct and I have no complaint.”

A “-" means :
"I think you are about 1/2 right", "There is something missing here, but not entirely wrong", "You missed the point somewhat, but what you did say was not exactly incorrect."

An “X” means:
"You did not answer the question", "The answer was too skimpy", "You basically missed the point".

A “?” means:
"I do not understand what you said", "Are you sure this makes sense?", "I think a part of what you said might not actually be true", "I wonder, I am not sure I believe you, but maybe you are correct".

A “+ +" means:
"I like what I see", "You went beyond the call of duty", "Nice job".

Homework grading should be thought of as a discussion, rather than a score. We mark up your work and return it to you. You can, and should if you feel it worth it, return your work with comments. I might then follow up in class or with additional comments, or you might follow up in class. In this way, a discussion takes place.