SYLLABUS

COSC-121, Fundamentals of Computer Systems
spring, 2015 (3 units)
Computer Science Department
Georgetown University

Prof. Richard K. Squier
St. Mary's, Room 339
202-687-6027
squier@cs.georgetown.edu

Lecture: Mon/Wed 14:00-15:15, Reiss 281

Office hours: Thu 16:00-17:00, or by appointment, or drop in.

TA: TBD

COURSE DESCRIPTION:

COSC-121 completes the picture of the fundamentals of computer systems begun in COSC-120 (Computer Hardware Fundamentals), focusing on the organization, performance, and low-level hardware-software interface and control of a complete computer system. We review instruction-set architectures (8086, LC4, and/or MIPS depending on time and interest), basic computing hardware concepts, and machine and assembly languages. We look at system software organization and interfaces from the bottom up, including the interactions of C code with assembly modules, the low-level control interfaces of some of the system hardware components (memory, keyboard, video display, and disk drive) developing low-level, interrupt-driven control software (assembly and C) in the context of system booting, operating system input/output services, and user program loading and execution. We survey the functional organization of the major system components and analyze performance effects of some refinements, including various types of processing parallelism (such as instruction-level parallelism, ILP, data parallelism, and task parallelism), the memory hierarchy, and I/O device communications. We survey major categories of modern high-performance computing systems and processors, including multi-cores, warehouse-scale computing, and the parallel computing taxonomy. Course work includes homework assignments, programming C with assembly language, and circuit design and system simulation using the Electric EDA and Verilog to modify and test a simple, MIPS-like processor. Prerequisite: COSC-120, Fundamentals of Computer Hardware. (Note, for spring, 2015, cosc-120 is not assumed to be a pre-requisite.)

Required text: Patterson and Hennessey. *Computer Organization and Design, 5th Edition.* ISBN: 978-0-12-407726-3. Morgan-Kauffman, 2013.
Suggested text: Harris and Harris. *Digital Design and Computer Architecture, 2nd ed.* ISBN 978-0-12394424-5. Morgan-Kauffman, 2012.
Suggested text: Patt and Patel. *Introduction to Computing Systems, 2ed.* ISBN-13 9780072467505. McGraw-Hill, 2004.

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 or exam 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. The ability to work cooperatively together is a learned skill that will be important later in life, and sharing information is central to that cooperation.

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, and not in determining grades (However, see class participation below.) I do use your submitted material as a guide in developing examinations. 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. Homework assignments, though checked for correctness, do not contribute towards your grade, except as class participation. As a consequence, there are no late penalties (but, no correction is done in that case).

Grading criteria: Class participation (attendance and homework submissions), 15%, midterm exam, 25%, project, 15%, final exam, 45%. In addition, extra credit of up to 100% of your other scores may be awarded for extended projects or other endeavors, provided prior instructor approval has been granted.

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. If your work is turned in late, we will not have the luxury of marking up your work; however, we will record that you turned it in, and that counts towards class participation.

Course Objectives (Don't be overwhelmed by this list)

Be able to:
Recognize hardware/software continuity and the hierarchy of virtual machines, and their languages;
Apply the power of abstraction in the context of virtual machines;
Understand the effects of AND, OR, NOT and EOR operations on binary data;
Represent data (characters, integers, and floating point) in digital form;
Estimate the magnitude of errors due to rounding effects and their propagation in chained calculations;
Understand basic digital representation of analog quantities and quantization errors;
Use register transfer language to describe internal operations in a computer;
Understand how a CPU’s control unit interprets a machine-level instruction;
Appreciate the concept of an instruction set architecture, ISA, the nature of machine-level instruction functionality and use of resources, and the relationship to micro-architecture;
Be aware of the various classes of instruction: data movement, arithmetic, logical, and flow control;
Recognize the difference between register-to-memory ISAs and load/store ISAs;
Understand the runtime stack: call/return, parameter passing, local work space creation and access;
Explain how interrupts are used to implement I/O control and data transfers;
Write low-level assembly language device handlers for keyboards and displays;
Write C language routines and hand link them to assembly language routines;
Explain in detail the actions of an assembler and its use of a symbol table;
Identify types of data communications and their performance ranges, and device access coordination;
Identify memory organizations and technologies and understand their performance effects;
Understand why a memory hierarchy is necessary to reduce the effective memory latency;
Describe the various ways of organizing cache memory and the cost-performance tradeoffs;
Appreciate the need for cache coherency in multiprocessor systems;
Describe pipelined processor performance and the effect of exceptions;
Understand system performance: processors, memory systems, buses, and software;
Describe superscalar's use of multiple functional units and instructions per cycle;
Understand computer performance measured in MIPS or SPECmarks and the limitations;
Appreciate the power/performance tradeoff and the need to minimize power consumption;
Discuss the concept of parallel processing and performance, including Flynn’s taxonomy;
Apply Amdahl’s law and Moore’s law and understand their consequences;
Understand basic organization of multi-core, many-core, and multi-threaded processors;
Understand basics of vector processing and the VLIW concept and its relationship to EPIC;
Explain the concept of branch prediction in enhancing the performance of pipelined machines;
Understand how speculative execution can improve performance;
Describe superscalar architectures and program correctness under out-of-order execution;
Explain the performance advantages that multithreading can offer and when it cannot help;
Identify the basic properties of bandwidth, latency, scalability and granularity;
Explain the purpose of snooping, directory-based coherency protocols, and sequential consistency;