SYLLABUS

COSC-121, Fundamentals of Computer Systems, spring 2012 (3 units)
Computer Science Department
Georgetown University

Lecture: 15:30-16:45 PM MW, White-Gravenor 211

Prof. Richard K. Squier
St. Mary's Hall, 339
phone: 202-687-6027
squier@cs.georgetown.edu
office hours: Mon/Wed 14:00-15:00

COURSE DESCRIPTION:
This course is a follow on to COSC-120 (Computer Hardware Fundamentals), and completes the picture of the fundamentals of computer systems. The focus is on the organization, performance, and low-level hardware-software interface and control of a complete computer system. First, we briefly review the instruction-set architectures of two CPUs (MIPS and LC3) and basic computing hardware concepts. We then develop some basic assembly language programming skills and apply those to the low-level control interfaces of some of the system hardware components; the keyboard, video display, and disk drive. We develop their low-level, interrupt-driven control software in the context of the input/output system software interface to the operating system. We also consider the functional organization of the major system components and analyze performance effects of some refinements, including 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. Course work includes weekly programming and homework assignments, and an implementation in C and assembly language of a bootable, micro-level operating system. Prerequisite: COSC-250, Fundamentals of Computer Hardware.

TEXTS:

Required: Patterson and Hennessey. *Computer Organization and Design, Revised 4th Edition*. ISBN: 978-0-12-374750-1. Morgan-Kauffman, 2012.

Recommended: 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. 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, 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, and may 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.

Grading criteria: Class participation (attendance and homework submissions), 15%, midterm exam, 15%, micro-OS project, 40%, final exam, 30%. In addition, or as a partial substitute for parts of the OS project, we may be able to build a finite-state machine using TTL chips and breadboards. Extra credit of up to 100% may be awarded for extra-ordinary 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 the continuity of hardware/software, the concept of a virtual machine, the hierarchy of virtual machines, and their intermediate 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 runtime stack content from how subroutines are called and return, how parameters are passed to subroutines, and how local work space is created and accessed;
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 various types of data communications in a computer system and their performance ranges, and understand 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 how processor performance is improved by overlapped instruction execution in pipelining, and the effect of exceptions;
Understand the difference between processor performance and system performance (i.e., the effects of memory systems, buses, and software on overall performance);
Describe superscalar's use of multiple functional units in executing multiple instructions per cycle;
Understand how computer performance is expressed by measurements such as MIPS or SPECmarks and the limitations of such measurements;
Appreciate the relationship between power dissipation and computer performance and the need to minimize power consumption;
Apply Amdahl's law and Moore's law and understand their consequences;
Understand basic organization of multi-core, many-core, and multi-threaded processors;
Be able to discuss the concept of parallel processing and the relationship between parallelism and performance, including Flynn's taxonomy for multiprocessor structures and architectures;
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 the need to ensure program correctness when executing instructions out of order;
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;