Description
We study the design and performance of a variety of computer systems from simple 8-bit micro-controllers through 32/64-bit RISC architectures all the way to ubiquitous x86 CISC architecture. We’ll start from logic gates and digital circuits before delving into arithmetic and logic units, registers, caches, memory, stacks and procedure calls, pipelined execution, super-scalar architectures, memory management units, etc. Along the way we’ll study several typical instruction set architectures and review concepts such as interrupts, hardware and software exceptions, serial and other peripheral communications protocols, etc. A number of programming projects, frequently done in assembly language and using various processor simulators, round out the course.

Prerequisites
Required: Introduction to Programming (EN.600.107 or equivalent).
Recommended: Intermediate Programming (EN.600.120 or equivalent).

Instructor
Peter H. Fröhlich, phf@cs.jhu.edu, http://gaming.jhu.edu/~phf/
Office: Malone 223, 410-516-8710
Office hours: Mondays & Fridays 11:30 am–12:30 pm, Wednesdays 3:00 pm–4:00 pm, and by appointment.

Teaching Assistant
Jose Nino, jnino1@jhu.edu
Office hours: Tuesdays 10:00 am–11:00 am in Ugrad lab, Tuesdays 10:15 am–11:15 am in Ugrad lab.
Additional course assistants: See http://piazza.com/jhu/spring2016/600233

Meetings
Monday, Wednesday, Friday, 1:30 pm–2:20 pm, Shaffer 100

Textbook
No required text.

Online Resources
http://gaming.jhu.edu/~phf/2016/spring/cs233/
http://piazza.com/jhu/spring2016/600233
Course Objectives

(1) Fundamentals of digital circuits and digital design.
(2) Basics of CPU (central processing unit) and ISA (instruction set architecture) design.
(3) Familiarity with a variety of representative CPU architectures and their assembly languages.
(4) Familiarity with topics in computer architecture (such as pipelining, branch prediction, caches, memory management) that impact performance of computer systems in general.

Course Topics

- Digital circuits and digital design.
- Basic CPU and ISA design.
- The MOS 6502 CPU (8 bit).
- Interrupts and exceptions.
- Subroutines and stack.
- The MIPS CPU (32 bit).
- Pipelined architectures.
- Branch prediction.
- Design and implementation of caches.
- The x86/AMD64 CPU (32/64 bit).
- Memory management unit.
- Linking and loading.

Course Expectations & Grading

Assignments (about 10): 70%, Midterm: 10%, Final: 20%. See http://gaming.jhu.edu/~phf/policies.html for details.

Key Dates
See http://gaming.jhu.edu/~phf/2016/spring/cs233/ for details.

Assignments & Readings
See http://gaming.jhu.edu/~phf/2016/spring/cs233/ for details.

Ethics

The strength of the university depends on academic and personal integrity. In this course, you must be honest and truthful, abiding by the Computer Science Academic Integrity Policy:

Cheating is wrong. Cheating hurts our community by undermining academic integrity, creating mistrust, and fostering unfair competition. The university will punish cheaters with failure on an assignment, failure in a course, permanent transcript notation, suspension, and/or expulsion. Offenses may be reported to medical, law or other professional or graduate schools when a cheater applies.

Violations can include cheating on exams, plagiarism, reuse of assignments without permission, improper use of the Internet and electronic devices, unauthorized collaboration, alteration of graded assignments, forgery and falsification, lying, facilitating academic dishonesty, and unfair competition. Ignorance of these rules is not an excuse.

Academic honesty is required in all work you submit to be graded. Except where the instructor specifies group work, you must solve all homework and programming assignments without the help of others. For example, you must not look at anyone else's solutions (including program code) to your homework problems. However, you may discuss assignment specifications (not solutions) with others to be sure you understand what is required by the assignment.
If your instructor permits using fragments of source code from outside sources, such as your textbook or on-line resources, you must properly cite the source. Not citing it constitutes plagiarism. Similarly, your group projects must list everyone who participated. Falsifying program output or results is prohibited.

Your instructor is free to override parts of this policy for particular assignments. To protect yourself: (1) Ask the instructor if you are not sure what is permissible. (2) Seek help from the instructor, TA or CAs, as you are always encouraged to do, rather than from other students. (3) Cite any questionable sources of help you may have received.

On every exam, you will sign the following pledge: "I agree to complete this exam without unauthorized assistance from any person, materials or device. [Signed and dated]". Your course instructors will let you know where to find copies of old exams, if they are available.

Report any violations you witness to the instructor. You can find more information about university misconduct policies on the web at these sites:

- Undergraduates: e-catalog.jhu.edu/undergrad-students/student-life-policies/
- Graduate students: e-catalog.jhu.edu/grad-students/graduate-specific-policies/

Students with Disabilities

Any student with a disability who may need accommodations in this class must obtain an accommodation letter from Student Disability Services, 385 Garland, (410) 516-4720, studentdisabilityservices@jhu.edu.

ABET Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline (a)
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution (b)
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs (c)
- Recognition of the need for and an ability to engage in continuing professional development (h)
- An ability to use current techniques, skills, and tools necessary for computing practice (i)
- An ability to apply design and development principles in the construction of software systems of varying complexity (k)