University of Utah
School of Computing

CS 4960, Section 2
History of Electronic Computing
Spring 2008

Course Description

CS 4960, History of Electronic Computing, is a three credit hour class offered by the School of Computing. There are no formal pre- or co-requisites; however, students should have some familiarity with the concepts of computer architecture, and some experience with at least one programming language. Here is brief description of the course and its goals:

> The sharp increase in speed, reliability, and usability of computers over the past 60 years is nothing short of miraculous. However, the notions of how people use computers, and even what a computer is, have shifted dramatically during that time. This course reviews the significant technologies that have shaped the computing industry since the mid-twentieth century, with special attention to the people and companies that pioneered these advances. Students will examine not only the hardware, but the significant operating systems, programming languages, and interaction techniques that defined computing throughout history. Along the way, this course will examine certain recurring themes and trends that have punctuated the computing industry since its beginnings. The goal of this course is to help students to apply the lessons learned from computing's past to today's engineering problems — giving them the perspective that will enable them to be more effective technological leaders in the future.

Contact Information

Instructor
Geoff Draper
Office: 2780 WEB
Office Hours: M/W 12:50-1:50
or by appointment.
(Office hours will be held
in the Linux CADE lab)
Email: draperg@cs.utah.edu

Teaching Assistant
Nathan Dykman
Office: 4142 MEB
Office Hours: Tu 3:15–5:15

Information and Questions
http://www.cs.utah.edu/classes/cs4960-02
teach-cs4960-02@cs.utah.edu

Lectures
Monday, Wednesday, Friday
11:50 a.m. – 12:40 p.m.
122 WEB

Departmental Contacts
School of Computing
3190 MEB
(801) 581-8224

Why Study Computing History?

The purpose of a university-level computer science education is not merely to train students to become professional “coders,” but to cultivate future *leaders* and *problem solvers*. Thus we study history not for its own sake, but for the lessons that can be learned by the successes *and* failures of past systems and architectures — and of the companies that made them.

Content Overview

Computers today are far more powerful than their counterparts 10, 25, or 50 years ago. The advances in speed, reliability, and ease-of-use are nothing short of phenomenal. Were these improvements solely the result of the drive of engineers who willed them into being? Or are they due to serendipity, of happenstance innovations happening at the right place and time? Or is the truth somewhat a mixture of these two extremes?

In this course, we will review the significant technologies that have shaped the computing industry since the mid-twentieth century. Our study will be roughly guided by chronology, examining most events in the order that they took place. However, throughout the course we will also follow an undercurrent of thematic topics and trends: What made Technology A superior to Technology B? Why did Company A fold while Company B prospered? *And most importantly, how can we use this knowledge to avoid the pitfalls of the past, and to influence the development of future technology for the better?*

Along the way, we will discover a number of recurring themes in the history of computing:

- **In the long run, software is more expensive than hardware.** This is one of the most puzzling paradoxes in computing, yet there are plenty of examples to support it. How often have you walked into an office and looked at the receptionist’s computer? Most of the time, it is a relatively new PC. Take a look at the software he or she is running. Surprisingly often, one of the windows on the receptionist’s desktop will be a terminal emulator running some ancient special-purpose text-based program that the firm acquired decades ago. (I know of a company that still actively recruits COBOL programmers. Apparently it cost less to maintain their existing programs than to rewrite them in a modern language!) Thus, backwards compatibility is a HUGE consideration for any new computer or operating system. If it can’t run people’s legacy applications, it faces an uphill battle for users’ acceptance.

- **Cheap general-purpose hardware beats expensive special-purpose hardware almost every time.** Remember when the Internet first became really popular? Folks were predicting the demise of the PC... all people *really* needed was an “Intenet appliance” with an email client and a web browser. Sounds great in theory, but the market never took off. Same with the ill-fated handheld “eBook readers.” This trend has repeated itself throughout history. Pixar used to make their own specialized hardware for producing CGI movies. Now they use commodity workstations. So-called “dumb terminals,” once touted as a cost-effective way of allowing multiple users access to “expensive” computing resources, gradually disappeared as computers themselves became less expensive than the terminals!

- **Most “new ideas” in computing are actually quite old.** The Internet did not appear out of nowhere in the 1990’s. Computer networking has been around since the 1960’s. The concept of hypertext, the basis of the World Wide Web, has been around even longer. Technologies such as multitasking, virtual memory, and hardware emulators – which are relatively new to personal computers, were commonplace in the mainframe and minicomputer systems of the 1960’s and 1970’s.

- **The court system is not the best place to resolve technological issues.** Back in IBM’s heyday, its competitors tried to halt its progress by taking it to court. Ironically, by the time the lawsuits finished, the technologies being disputed were obsolete and the case was irrelevant. We can see similar patterns in the recent Microsoft and SCO lawsuits.

- **A poor standard is better than no standard at all.** Despite MS-DOS's (and later, Windows') flaws, the fact that it ran on so many different computers gave the software industry something they could throw their support behind. We see the same thing in the Fortran and COBOL programming languages. Although not perfect, they were supported on a variety of systems and thus rose to prominence.

- **New breakthroughs are still possible. Not all the "low-hanging fruit" is gone!** In the IBM antitrust trial of the 1980's, one expert testified that no new major innovations in computing were likely. This was spoken before the Internet became popular! Yet with each new advance in technology, predictions are often made that the new technology will shorten the workday, decrease unemployment, solve world hunger, etc. Alas, we have not yet reached the long-promised "Digital Utopia." Technology has undoubtably improved the quality of life for mankind, but not always in the way its creators envisioned it.

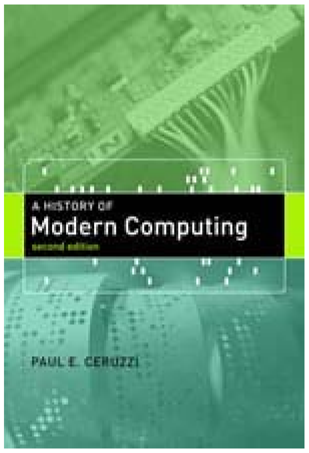
Course Learning Objectives

In this course, you will learn about the scientific and engineering innovations that have guided the development of computing technlogy from the mid-twentieth century to today. You will also learn about the pioneering individuals and organizations that developed these technologies. You will examine the reasons why some of these organizations still flourish today, while others faded into obscurity. When you complete this course, you should be able to:

- Explain the advantages and disadvantages of the stored-program (von Neumann) architecture.
- Explain the evolution from vacuum tubes to transistors to integrated circuits, and the effect this evolution had on the cost, performance, and reliability of computers.
- Explain the definining characteristics of a mainframe, minicomputer, and personal computer. Identify the historically significant machines from each of these eras.
- Formulate an argument regarding the influences, both positive and negative, of the GUI on computer usability.
- Install and use an emulator for an obsolete computing platform.
- Write a simple program in a "historic" programming language.
- Write a simple program for an obsolete architecture, and run it via an emulator.
- Explain the engineering principles behind early memory systems such as mercury delay line memory, magnetic drum memory, and magnetic core memory.
- Formulate an argument as to why some early computer companies (such as IBM) were able to adapt and prosper in a changing commercial market, while others (such as DEC) did not.
- Identify the core concepts from early programming languages that are still in use in modern programming languages.
- Observe trends in today's commercial computing marketplace that mirror those of the twentieth century.

Course Organization

Reading. The required text is A History of Modern Computing, Second Edition by Paul E. Ceruzzi.



Grading. There are 1000 points available this semester, allocated thus:

```
700 points from assignments
100 points from midterm exam
100 points from final exam
60 points from class participation/citizenship
40 points from "daily test questions"
----------------------------
1000 points total
```

Final grades for the course will follow this approximate distribution:

- **A** : 930 points and above
- **A-** : 900 – 929 points
- **B+** : 870 – 899 points
- **B** : 830 – 869 points
- **B-** : 800 – 829 points
- **C+** : 770 – 799 points
- **C** : 730 – 769 points
- **C-** : 700 – 729 points
- **D+** : 670 – 699 points
- **D** : 630 – 669 points
- **D-** : 600 – 629 points
- **E** : 599 points and below

Please note that the instructor reserves the right to modify the grading scale in favor of the students as deemed advisable at the end of the semester.

Class Participation. Class meets three times a week for 50 minute lectures. Lectures consist primarily of demonstrations, discussions, and guest speakers — an unusual format for a computer science class. For this to work, however, you need to be both physically and mentally in the classroom. A small percentage of your overall grade is based on your attendance in class. You will receive additional points for participating in class. Examples of participation include, but are not limited to: answering questions in class, asking questions in class, contributing meaningfully to classroom discussion, and emailing questions/comments to the instructor. There are 60 points possible in this category. You earn points when I feel that you are participating, and you lose points when I notice your attention drifting (i.e. using a laptop or cell phone in class is a quick way to lose points).

"Daily Test Questions" At the end of every class session, you will write and turn in a "test question" that captures one of the main points of that day’s discussion. You will not be graded on the content of these test questions, but your participation in them will count as a small percentage of your overall grade. Their main purpose is to help the instructor assess your understanding of the course material in a safe, low-stress way. You earn one point point for each lecture in which you turn in a test question, with a total of 40 points available. If I choose one of your questions to appear on an actual exam, then you earn bonus points!

Meeting with the Instructor. At least once during the semester, you are requested to schedule an appointment to meet with the instructor one-on-one. Typically, this will be in the CADE lab. These brief interviews will allow the instructor to get to know you better as an individual, and not just another "face in the crowd." These interviews will also allow you to give the instructor candid feedback on how he can better tailor the course to suit your personal and professional goals. Participation in this interview counts as an assignment, and therefore figures as small percentage (3%) of your overall grade.

Homework Assignments. There will be a handful of homework assignments. Many will be small, like decoding an encrypted message, or writing a one-page paper on a specific aspect of computer history. There will also be a couple of longer assignments, like the term paper and the final project. Details on each homework assignment can be found on the assignments page.

Late Policy. Assignments must be turned in by the posted deadline to be eligible for full credit. Assignments may be turned in up to 24 hours late for a 10% penalty. Assignments will not be accepted later than 24 hours after the deadline. On the other hand, everyone's situation is different, so if you need an exception to this policy, please discuss it with the instructor or T.A. BEFORE the assignment deadline.

Exams. There will be a midterm exam and a final exam (during finals week). You will be allowed to bring one page of notes to the midterm and two pages of notes to the final. The midterm will cover material covered during the first half of the semester, and the final will cover material from the entire semester.

Getting Help and Information

The class web page is http://www.cs.utah.edu/classes/cs4960-02. It will contain a variety of information resources, including course staff consulting hours and e-mail addresses, answers to frequently asked questions, handouts, and assignments.

You should subscribe to the class mailing list (cs4960-02@cs.utah.edu). The instructor and TA will use this list to send you important information throughout the semester.

There is a staff mailing list (teach-cs4960-02@cs.utah.edu). The instructor and TA will receive a copy of every message that is sent to this mailing list. We will reply directly to each question, and we will post the answers to frequently asked questions to the class web page. We encourage you to seek us out whenever you need help, advice, or encouragement. We will always be available during our regular office hours, and you can make appointments for other times. Simple questions can often be answered by e-mail. Our consulting schedule will be posted on the class web page as soon as it is finalized.

Topical Outline

Here is an outline of some of the topics we will be covering this semester. The exact arrangement of these topics into dates and lectures will be determined by the pace at which classroom discussion proceeds.

1. Early mechanical or electronic computers
    - Abacus
    - Babbage
    - Zuse
    - Enigma

    - ENIAC
2. The Advent of Commercial Computing (1945-1956)
    - Stored program architecture
    - UNIVAC
    - IBM 701
3. Computing Comes of Age (1956-1964)
    - Transistors replace vacuum tubes
    - Core memory replaces drum memory
    - Special purpose computers trumped by general purpose stored program computers
    - Disk storage
4. Early History of Software (1952-1968)
    - “plan preparation machine”
    - “algebraic system” translates equations to machine code
    - assembly language
    - FORTRAN
    - COBOL
    - operating systems
    - computer science as the study of algorithms
    - structured programming / software engineering
5. From Mainframe to Minicomputer (1959-1969)
    - Data entry to punched cards to mainframe... not interactive, but suggestive of it.
    - minicomputers used shorter word length (cheaper hardware) but used programming hacks to get around it.
    - minicomputers were interactive: they set the stage for personal computers.
    - minicomputers opened up computing to markets that had never considered using computers before.
    - DEC
6. The “Go-Go” Years and System/360 (1961-1975)
    - Snow White and the 7 Dwarfs
    - BUNCH
    - IBM System/360: a line of compatible computers
        - microprogramming allowed machines with different speeds to have compatible instruction sets.
        - emulators/VMs gave backwards compatibility - software is more permanent and hard to modify than hardware!
        - 32-bit word length; introduction of “bytes"
        - EBCDIC va ASCII
    - customers wanted time-sharing; IBM was slow to respond.
        - System/370 in 1970.
    - attack of the clones - not just whole computers, but cheap replacement parts (plug compatible manufacturers)
    - software vendors
7. The Chip and its impact (1965-1975)
    - Made transistors faster and cheaper and smaller
    - Greatly influenced by “space race"
    - standardization
    - PDP-11
    - BASIC and computer science education
    - minicomputer sowed “the seeds of its own destruction"
8. The Personal Computer (1972-1977)
    - Intel 4004, 8008, finally 8080

    - Altair 8800 opened the floodgates
    - Microsoft BASIC
    - CP/M operating system
9. Augmenting Human Intellect (1975-1985)
    - DEC VAX
    - Improved terminals
    - Word Processing
    - Xerox PARC and GUIs
    - Personal Computers, Part Deux. These all came with monitors, keyboards, and tape (or floppy disk) storage. At last computing was opened up for non-hobbyists.
        - TRS-80 (1977)
        - Commodore PET (1977)
        - Apple II (1977)
    - VisiCalc - the killer app that made people WANT computers
    - IBM PC (1981) and MS-DOS
    - standardization!
    - Macintosh
    - portable computers (Osborne and TRS80 model 100)
10. Workstations, UNIX, and the Net (1981-1995)
    - Workstation: more powerful than a PC, less expensive than a minicomputer (SUN and HP)
    - Ethernet
    - DEC falls
    - CISC versus RISC
    - Ethernet
    - Internet
    - World Wide Web and Mosaic
    - Netscape
11. Internet Time (1995-present)
    - IE 4.0 and Microsoft antitrust lawsuit
    - Internet
    - Java - interactive web pages
    - dot-com fallout; eBay and Amazon
    - Google
    - Linux
12. Miniaturization
    - Palm
    - cell phones
    - iPod
    - Blackberry

ADA Statement

The University of Utah seeks to provide equal access to its programs, services and activities for people with disabilities. If you will need accommodations in the class, reasonable prior notice needs to be given to the Center for Disability Services, 162 Union Building, 581-5020 (V/TDD). CDS will work with you and the instructor to make arrangements for accommodations.

Rights and Responsibilities

All students are expected to maintain professional behavior in the classroom setting, according to the Student

Code, spelled out in the Student Handbook. Students have specic rights in the classroom as detailed in Article III of the Code. The Code also species proscribed conduct (Article XI) that includes cheating on tests, plagiarism, and/or collusion, as well as fraud, theft, etc. Students should read the Code carefully and know they are responsible for the content. According to Faculty Rules and Regulations, it is the faculty responsibility to enforce responsible classroom behaviors, and I will do so, beginning with verbal warnings and progressing to dismissal from class and a failing grade. Students have the right to appeal such action to the Student Behavior Committee.