15-740: Computer Architecture
Spring 2018
Syllabus

1 Course Details at a Glance

Lectures: Tuesdays and Thursdays, 3:00-4:20pm, GHC 4303

Instructor: Nathan Beckmann, GHC 9021, `beckmann@cs.cmu.edu`
Office Hours: Thursdays, 4:30-5:30pm, or by appointment

TAs: Sivaprasad Sudhir, `sivapras@andrew.cmu.edu`
Office Hours: TBD

Web Page: `www.cs.cmu.edu/afs/cs/academic/class/15740-s18/www/`
Course Materials: `/afs/cs.cmu.edu/academic/class/15740-s18/public`
Course Announcements: will be posted on the web

2 Textbooks

There is no required textbook for the course. Most lectures have accompanying papers you are expected to read for background material. However, the following two texts are useful and will be referenced.

Main Text: Hennessy, J. L, and Patterson, D. A., *Computer Architecture: A Quantitative Approach, 5th Edition*. Morgan Kaufmann, 2011.

Supplemental Text: Culler, D., and Singh, J.P., with Gupta, A. *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, 1998.

3 Course Overview and Objectives

This course attempts to provide a deep understanding of the issues and challenges involved in designing and implementing modern computer systems. Our primary goal is to help students become more skilled in their *use* of computer systems, including the development of applications and system software. Users can benefit greatly from understanding how computer systems work, including their strengths and weaknesses. This is particularly true in developing applications where performance is an issue.

In addition to covering issues that are relevant in systems with only a single processor, the course places a strong emphasis on *parallel systems* comprised of multiple processors, with topics ranging from programming models to hardware realizations. While the Hennessy and Patterson textbook covers nearly all of the topics that will be discussed in this course, the Culler, Singh and Gupta textbook goes into greater depth on parallel architecture and programming.

3.1 Course Themes

An addition to our "user-centric" (vs. "builder-centric") approach, the course has several other themes. One theme is to emphasize the role of evolving technology in setting the directions for future computer systems. Computer systems, more than any other field of computer science, has had to cope with the challenges of exploiting the rapid advances in hardware technology. Hardware that is either technologically infeasible or prohibitively expensive in one decade, such as bitmapped full color displays or gigabyte disk drives, becomes consumer products in the next. Technology that seems to have a bright future, such as magnetic bubble memories, never becomes competitive. Others, such as CMOS, move from being a niche technology to becoming dominant. In addition, computer systems must evolve to support changes in software technology, including advances in languages and compilers, operating systems, as well as changing application requirements. Rather than teaching a set of facts about current (but soon obsolete) technology, we therefore stress general principles that can track evolving technology.

Another theme of the course is that "hands-on" exercises generally provide more insight regarding system behavior than paper-and-pencil exercises. Hence our assignments involve programming and using computer systems, although in a variety of different ways.

Finally, rather than stopping with state-of-the-art in computer architecture as of a decade ago, another theme of this course is looking at the state-of-the-art today as well as open research problems that are likely to shape systems in the future. Hence we will be discussing recent papers on architecture research in class, and students will perform a significant research project.

4 Prerequisites

This course is not intended to be your first course on computer architecture or organization; it is geared toward students who have already had such a course as undergraduates. For example, we expect that people are already at least somewhat familiar with assembly language programming, pipelining, and memory hierarchies. If you have not had such a course already, then it is still possible to take this course provided that you are willing to spend some additional time catching up on your own. If you feel uncertain about whether you have adequate preparation, please discuss this with the instructor.

In addition to an undergraduate computer organization course, here are some other topics which are helpful for this course (references are included for self study):

- A working knowledge of C, including pointers and memory allocation [Kernighan+88].
- An introductory compiler course, especially aspects of code generation such as data formats, procedure linkages, and translation of control constructs [Aho+86, Chs. 1–2, 7–9].
- An introductory operating system course, especially aspects of protection, scheduling, and concurrency [Silberschatz+91, Chs. 4.1–4.3, 5.1–5.4, 5.7, and 7–9]
- Bit-level representation of and manipulation of numbers [HenPat96, App. A.1–A.2].

5 Course Work

Grades will be based on homeworks, a research project, two exams, and class participation.

Homeworks: There will be two homework assignments. Each assignment involves a non-trivial amount of programming. You will work in groups of two or three (preferably three) people on the assignments. (Turn in a single writeup per group.)

Project: A major focus of this course is the project. We prefer that you work in groups of two on the project, although groups of up to three may be permitted depending on the scale of project (ask the instructor for permission before forming a group of three). The project is intended to be a scaled-down version of a real research project. The project must involve an experimental component—i.e.,

it is not simply a paper and pencil exercise. We encourage you to come up with your own topic for your project, although we can give you suggestions if you are stuck. You will have six weeks to work on the project. You will present your findings in a written report (the collected reports may be published as a technical report at the end of the semester), and also during a poster session during the last day of class. Start thinking about potential project ideas soon!

Exams: There will be two exams, each covering its respective half of the course material. Note that the second exam is not cumulative, and is weighted equally with the first exam. Both exams will be closed book, closed notes.

Class Participation: In general, we would like everyone to do their part to make this an enjoyable interactive experience (one-way communication is no fun). Hence in addition to attending class, we would like you to actively participate by asking questions, joining in our discussions, etc. Three classes are set aside entirely for student-led in-class discussions on active areas of research and innovation in computer architecture. All students are expected to lead one of these discussions.

5.1 Grading Policy

To pass this course, you are expected to demonstrate competence in the major topics covered in the course. Your overall grade is determined as follows:

Exams:	30% (15% each)
Assignments:	10% (5% each)
Readings:	10%
Project:	40%
Class Participation:	10%

Late assignments will not be accepted without prior arrangement.

References

[Aho+86] Aho, A. V. and Sethi, R. and Ullman J. D., *Compilers*. Addison-Wesley, 1986. Background.

[CulSin98] Culler, D., and Singh, J. P., with Gupta, A., *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufman, 1998.

[HenPat96] Hennessy, J. L, and Patterson, D. A., *Computer Architecture A Quantitative Approach, 5th Edition*. Morgan Kaufman, 2011. Main textbook.

[Kernighan+88] Kernighan, B. W., and Ritchie, D. M., *The C Programming Language, 2nd edition*, Prentice Hall, 1988. Background.

[Silberschatz+91] Silberschatz, A., Peterson, J., and Galvin, T., *Operating System Concepts, 3rd Edition*, Addison-Wesley, 1991. Background.