

15-740: Computer Architecture

Fall 2000

Syllabus

1. Course Details at a Glance

Lectures:	Mondays, Wednesdays & Fridays, 10:30-11:59 a.m., NSH 1305
Instructor:	Todd C. Mowry, WeH 8123, 268-3725, tcm@cs.cmu.edu
TA:	Angela Demke Brown, WeH 3711, 268-1557, demke+@cs.cmu.edu
Class Administrator:	Maury Burgwin, WeH 8124, 268-4740, mburgwin@cs.cmu.edu
Web Page:	www.cs.cmu.edu/afs/cs/academic/class/15740-f00/www/
Newsgroup:	cmu.cs.class.cs740
Course Materials:	/afs/cs.cmu.edu/academic/class/15740-f00/public

2. Textbooks

Main Text: Hennessy, J. L, and Patterson, D. A., *Computer Architecture: A Quantitative Approach, 2nd Edition*. Morgan Kaufmann, 1996. (*NOTE: don't try to get by with the first edition of this book; the second edition is much different and more aligned to the content of this course.*)

Supplemental Text: Culler, D., and Singh, J.P., with Gupta, A. *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, 1998. (*Copies should be available in the E&S Library by the time we need them.*)

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.

The course material is divided evenly into two parts. The first half of the course covers systems based on a single processor, closely following the Hennessy and Patterson textbook. The second half of the course covers parallel systems containing multiple processors, with topics ranging from programming models to hardware realizations. The material for this latter half of the course can be found to some extent in the Hennessy and Patterson book, but is treated in much greater detail in the Culler, Singh and Gupta text.

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. Non-CS Students

If you are not a graduate student in the Computer Science PhD program, you need permission to take this class. If you have not already done so, send a message to the instructor stating your status, why you want to take the class, and if you want to take the class for credit or as an auditor.

6. Placing Out

Students who have already taken *graduate-level* courses in computer architecture or parallel architecture may find that some of this course material is familiar. Although the course topics (especially in the first half of the course) may look familiar even to students who have taken an undergraduate computer architecture course, this course is designed to build on undergraduate material, and will cover this topics in much greater depth. It is likely that the focus and style of this course will be different from what you have experienced before, and that the pace will be fast enough that you will not be bored. However, if you feel strongly that you should be able to “place out” of all or part of this course, contact the instructor.

7. Course Work

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

Homeworks: There will be roughly two homework assignments. Each assignment involves a non-trivial amount of programming. You may work in groups of up to 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 will be posting suggested projects to the class web page at a future date. 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). 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.

7.1. 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:	40% (20% each)
Homework:	15%
Project:	35%
Class Participation:	10%

Late assignments will not be accepted without prior arrangement.

8. Schedule

Table 1 shows the tentative schedule. There might be some variations.

Table 1: CS 740, Fall 2000, Tentative Schedule.

Class	Date	Day	Topic	Reading	Homework	Who
1	9/13	Wed	Performance & Technology	H&P Ch. 1	#1 Out	TCM
2	9/15	Fri	Alpha Programming	H&P Ch. 2		TCM
3	9/18	Mon	Instruction Set Comparison	H&P App. C & D		TCM
4	9/20	Wed	Basic Pipelining	H&P Ch. 3		TCM
5	9/22	Fri	Advanced Pipelining	H&P Ch. 4	#1 Due, #2 Out	TCM
6	9/25	Mon	Superscalar Processing Concepts	H&P Ch. 4		TCM
7	9/27	Wed	Superscalar Processing Practice	H&P Ch. 4		TCM
8	9/29	Fri	The Memory Hierarchy	H&P Ch. 5		ADB
9	10/2	Mon	Programming for Performance			ADB
10	10/4	Wed	Virtual Memory	H&P Ch. 5		ADB
11	10/6	Fri	<i>Recent Research on Uniprocessors I</i>	Handouts	#2 Due	N/A
12	10/9	Mon	<i>Recent Research on Uniprocessors II</i>	Handouts		N/A
	10/11	Wed	Exam I			
13	10/13	Fri	Intro to Parallel Architecture	H&P Ch. 8, CSG Ch. 1		TCM
14	10/16	Mon	Parallel Programming I	CSG Ch. 2		TCM
15	10/18	Wed	Parallel Programming II	CSG Ch. 3 & 4		TCM
	10/20	Fri	<i>Mid-Semester Break (No Class)</i>		Project Proposal	N/A
	10/23	Mon	<i>Mid-Semester Break (No Class)</i>			N/A
16	10/25	Wed	Cache Coherence I	CSG Ch. 5		TCM
17	10/27	Fri	Cache Coherence II	CSG Ch. 5		TCM
18	10/30	Mon	Memory Consistency	CSG Ch. 6		TCM
19	11/1	Wed	Synchronization	CSG Ch. 5		ADB
20	11/3	Fri	Interconnection Networks	CSG Ch. 10		ADB
21	11/6	Mon	<i>Recent Research on Multiprocessors</i>	Handouts		N/A
	11/20	Mon			Project Milestone	
	11/29	Wed	Exam II			
	12/4	Mon			Project Due	
	12/11	Mon	<i>Project Poster Session</i>			

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, 2nd Edition*. Morgan Kaufman, 1996. 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.