ENEE446 Digital Computer Design, Spring 2022

Lectures: MW 5:00 - 6:15.

Instructor: Dr. Uzi Vishkin
Telephone: 301-405-6763
E-mail: vishkin@umd.edu
Office: Iribe 5216
Office hours: M 3:00 - 4:00, by appointment.

Teaching Assistant (TA): Mr. Alex Mattingly
E-mail: amatti@terpmail.umd.edu
Office hours venue: TBD
Office hours: TBD.

Text: J.L. Hennessy and D.A. Patterson. Computer Architecture a Quantitative Approach, Sixth (!) Edition. Morgan-Kaufmann, San Francisco, 2017.
References for some of the enrichment material:
– U. Vishkin. Using simple abstraction to reinvent of computing for parallelism. Communications of the ACM 54,1 (January 2011), 75-85.
– How parallelism could look from the algorithms/programmer side. U. Vishkin, Thinking in parallel, http://www.umiacs.umd.edu/users/vishkin/PUBLICATIONS/classnotes.pdf

Prerequisite: A course in computer organization.

Prerequisite knowledge: Students should be familiar with at least one assembly language and one high level programming language such as C. The **first midterm exam** will be on Wednesday, March 9, during class.

The **second midterm** will be Monday, May 2, during class. There is no final exam.

Course objectives The main objective of the course is to introduce the basic concepts in contemporary computer architecture, including instruction sets, pipelining, advanced processor design, memory hierarchy, and storage systems. The emphasis will be on quantitative evaluation of various design issues. Specific examples from current microprocessors will be given.

Course description The course will cover the following: Principles of com-

puter design; cost/performance of design options; processor design. instruction set design and implementation; pipelining and instruction-level parallelism; floating-point arithmetic; memory-hierarchy design; caches; main memory; virtual memory; input/output design and performance measures; types of I/O devices; connections of I/O to CPU and main memory; last, but not least: the increasing role of parallelism from fine-grained to coarse-grained, what forms of parallelisms appear to be easier for programmers and the often missed critical importance of bandwidth (among processors, or between processors and memories).
The text is the basis for the course and is the backbone for the organization of the course. In particular, **no class is over till you went home, read the material in the text, and reached a critical understanding of it**. Please see more on the teaching approach later.

Grading
Up to 15% quizzes whose date will not be preannounced.
Up to 20% for Q&A written homework. Each student is required to submit at least 80% of the Q&A written homework and all projects.
Up to 10% for each project.
Up to 25% first midterm exam.
Up to 35% second midterm exam. If your grade for the second midterm is higher than your first midterm grade, the second midterm grade can be up to 50% of your final grade.
Note: The weight for quizzes will reduce the weight for other components, but will keep the ratio between them.

Tentative schedule
Chapter 1: weeks 1-2.
Chapter 2: weeks 2-3.
Chapter 3: weeks 4-6.
Chapter 4: weeks 7-8.
Chapter 5: weeks 9-10.
Chapters 6-7: weeks 11-12.
Parallelism from the algorithms/programmer side: weeks 13-14.
While students will be responsible for the material in the first 3 appendices in the textbook, some of this background material will be covered in class.

Enrichment material: Some enrichment material will be covered.
Coverage of background and enrichment materials may interfere with the tentative schedule above.

Documented disability

If you have a documented disability and wish to discuss academic accommodations with me, please contact me as soon as possible.

A textbook-centered teaching approach - a rationale

Objective of course The main objective of the course is to reach a *critical understanding* of the textbook information by the students.
It is also a goal of the course to improve your ability to understand such information on your own, in the future.

Background We introduced this particular textbook to ENEE446 since: (i) it is very informative in conveying a relatively timely picture of the state-of-the-art of computer architecture, and (ii) it is commonly used at the top computer science and engineering Programs.
However, I find that some parts of the textbook are unusually verbose and overly opinionated. Also, the level of detail exceeds what students in 400-level courses need to get from a lecture. Instead, your prior training is sufficient for overcoming this low-level on your own.

The main objective for the classroom presentation is to overcome these hurdles towards the main objective of the course. It is particularly challenging for a reader to locate the "punch lines". Also, students need to be equipped with a lot of background data and dissenting opinions in order to reach a point where they will be able to critically evaluate for themselves opinions expressed in the book, and form an independent opinion (or at least understand the pros and cons of the various approaches).

To get the most out of this teaching method students need to remember:
No class is over till you went home, read the material in the text, and reached a critical understanding of it. It is also expected that you prepare for class by reading the material that will be discussed.

You may wonder: *who uses similar methods?*
Quite a few liberal arts programs are based on studying directly from sources, such as the original texts written by great thinkers. There, understanding texts is the main thing and the role of a teacher, or tutor, is to get the students to derive as much as possible from the text. The teacher's goal is to add to the text, but at the same time not "temper" with the access of the students to the source itself. Adding needed background, and questioning

of assertions, are often more important to the students than helping reading the text, which they can often do reasonably well on their own. Indeed, this undergraduate/graduate course: (i) tries to convey trends in computer architecture (which is always a debatable issue), (ii) covers an unusually broad range of issues, and (iii) requires different background discussions in the context of its various chapters.

As a final motivation, I would like to note that learning how to think and process content is as much a goal of university education as absorbing content. As an instructor, I will operate as a knowledge agent more than a content supplier (a role played by the text, and other literature, including on the Web). Graduate study (and work as professional engineers) is often a do-it-yourself process; however, you can, and should, expect help and guidance.

The chair of our department distributed a while ago an article entitled Thoughts on Teaching our Engineers-To-Be, by MIT Professor Warren Seering. The article points out that: Frequently students express serious reservations and sometimes great displeasure when faced with an engineering problem that requires clarification of objectives, synthesis, the making of assumptions, reduction to a form that can be solved, and interpretation of the result.

However, the main point of the article is that teaching how to cope with such an open-ended engineering problem may be the single most neglected issue in current engineering education. My understanding of the educational objectives of ENEE446 is that it should play a key role in educating computer engineering students to cope with open ended problems through the process of digesting the textbook, as will be demonstrated in class. My request from you is to approach this with an open mind, and realize that the reservations and displeasures you may have may be neither your fault NOR MINE. Just take it easy and do your best to get the most out of this important class.

Lastly, it may also be informative for you to know that the other part of my job as a Professor is doing research; there, I am also trying to be a content producer.