EE 5123 --- Computer Architecture (Spring 2016)
Course Syllabus

Instructor:
Lide Duan
Assistant Professor
Department of Electrical and Computer Engineering
University of Texas at San Antonio
http://engineering.utsa.edu/~lideduan/index.html

Office: AET 2.376
Phone: 210-458-5208
Email: lide.duan@utsa.edu
Class Meeting Time: 6:00-7:15PM (Mondays and Wednesdays)
Class Location: EB 2.04.02
Office Hours: 4:00 – 6:00PM (Mondays and Wednesdays) or by appointment.
Website: All class materials will be handled electronically through Blackboard Learn.

Course Description:
Computer architecture fundamentals, instruction set architectures, memory and cache hierarchy, microprocessor pipelining, instruction-level parallelism, data-level parallelism, thread-level parallelism, and request-level parallelism.

Prerequisite:
Graduate standing. Non-experience in computer architecture is assumed. However, course projects will require programming skills in high-level languages (e.g., C, C++, Java, Python).

Textbook:
Computer Architecture: A Quantitative Approach, 5th Edition. (By John Hennessy and David Patterson).

Course Topics (tentative):
- Fundamentals of computer architecture
- Instruction set architectures (ISA)
- Cache and memory hierarchy design
- Pipelining
- Instruction-level parallelism (ILP)
    - Branch predictions, dynamic scheduling, multiple issue and static scheduling, speculations, compiler techniques, ILP limitations, etc.
- Data-level parallelism (DLP)
    - Vector architectures, SIMD, GPUs, loop-level parallelism.
- Thread-level parallelism (TLP)
    - Centralized shared-memory, distributed shared-memory, multiprocessor memory coherence, synchronization, consistency, etc.
- Request-level parallelism (RLP)
    - Warehouse-scale computers, cloud computing.

Grading Policy:

- Homework assignments: **30%**
  - 3 assignments of 10% each
- Exams: **35%**
  - Exam 1 (10%) + Exam 2 (10%) + Final (15%)
- Project: **30%**
  - Project 1 (15%) + Project 2 (15%)
- Quizzes: **5%**
  - The instructor will randomly choose a few classes to ask the students to turn in an answer to a very simple question. These are mainly for checking attendance.
- Total: **100%**

About the Grading

- The final letter grades will be curved based on the ranks and score gaps.
- After the grade of each assignment/exam/project/quiz is posted, you have up to a week to see me for any misgrading or miscalculation. After that, the grade is finalized.

About the HW Assignments

- Tentatively, the 3 assignments will be out in the 4^{th}, 9^{th}, and 15^{th} week of the semester, and due in about one week.
- All must be turned in through Blackboard Learn (in pdf or Word) before the specified deadlines. No late turn-ins.
- HW questions are good examples of exam questions.

About the Exams

- Tentatively, Exam 1 will be on Monday 2/15 (class time); Exam 2 will be on Monday 3/28 (class time); and Final Exam will be on Monday 5/9 (6PM).
- All exams are open-books and open-handouts.
- No make-up exams (except for extremely special situations with legitimate proof and under discretion of the instructor).

About the Projects:

- There will be two projects. Both need programming.
- Tentatively, the two projects will be out in the 3^{rd} / 11^{th} week, and due in the 9^{th} / 16^{th} week, respectively.
- Project 1 will be a disassembler of the MIPS binary code. You will be given a MIPS binary file (in txt format), and need to disassemble it into the corresponding MIPS assembly code.
- Project 2 will be a cache performance simulator. You will be given a txt file that has a list of memory addresses, and need to simulate the cache behavior based on specified parameters (e.g., cache size, block size, associativity, replacement policy).
- The projects can be implemented in any high-level language (e.g., C, C++, Java, Python, etc.) under any operating system (e.g., Linux, Mac OS, Windows, etc.).
- **Both projects must be completed individually**. There's zero-tolerance on copying source code, from either your classmates or students who took this course before. Automatic source code checking tools will be used to check for coding plagiarism. Both the copier and copiee will lose all the points for the project. So, protect your source code!