ECE 4750 Computer Architecture, Fall 2015

Course Syllabus

School of Electrical and Computer Engineering
Cornell University

revision: 2015-09-25-15-19

1. Course Information

Cross Listed	CS 4420 Computer Architecture		
Prereqs	ECE 3140 (cross listed as CS 3420) or CS 3410		
Instructor	Prof. Christopher Batten, 323 Rhodes Hall, `cbatten@cornell.edu` Office Hours: 323 Rhodes Hall, Tuesday, 3:30–5:00pm		
Admin. Assistant	Daniel Richter, 314 Rhodes Hall, `tdr27@cornell.edu`		
Graduate TAs	Moyang Wang	`mw828`	Office/Lab Hours: 314 Phillips, Thu, 7:30–10:00pm
	Taejoon Song	`ts693`	Office/Lab Hours: 314 Phillips, Mon, 7:00–8:30pm
	Jason Setter	`jls548`	Office/Lab Hours: 314 Phillips, Tue, 7:30–10:00pm
	Wei Geng	`wg79`	Office/Lab Hours: 314 Phillips, Wed, 7:30–10:00pm
Undergraduate TA	Vasudharini Mannar `vkm22`		
Lectures	255 Olin Hall, Monday and Wednesday, 2:55–4:10pm		
Disc. Section	203 Phillips Hall, Friday, 2:30–3:20pm		
Required Materials	J. L. Hennessy and D. A. Patterson "Computer Architecture: A Quantitative Approach" 5th edition, Morgan Kaufmann, 2012 Cornell bookstore (new: $90, used: $68), Kraftees (used: $65), Amazon ($60) On reserve online as an e-book and in Uris Library as a hard copy		
	D. M. Harris and S. L. Harris "Digital Design and Computer Architecture" 2nd edition, Morgan Kaufmann, 2012 Cornell bookstore (new: $90, used: $68), Amazon ($72) On reserve online as an e-book and in Uris Library as a hard copy		
	"ECE 4750 Course Packet" Available at Cornell bookstore ($55) or on reserve in Uris Library Selected portions will be available electronically		
Website	`http://www.csl.cornell.edu/courses/ece4750`		
Staff Email	`ece4750-staff@csl.cornell.edu`		

2. Description

This course aims to provide a strong foundation for students to understand modern computer system architecture and to apply these insights and principles to future computer designs. The course is structured around the three primary building blocks of general-purpose computing systems: processors, memories, and networks.

The first half of the course focuses on the fundamentals of each building block. Topics include instruction set architecture; single-cycle processors; hardwired vs. microcoded FSM processors; pipelined processors; direct-mapped vs. associative caches; pipelined caches; network topology, routing, and flow control; and integrating processors, memories, and networks. The second half of the course delves into more advanced techniques and will enable students to understand how these three building blocks can be integrated to build a modern shared-memory multicore system. Topics include superscalar execution, out-of-order execution, register renaming, memory disambiguation, branch prediction, and speculative execution; multithreaded, VLIW, and SIMD processors; non-blocking cache memories; memory protection, translation, and virtualization; and memory synchronization, consistency, and coherence. Students will learn how to evaluate design decisions in the context of past, current, and future application requirements and technology constraints.

This course includes a significant project decomposed into five lab assignments. Throughout the semester, students will gradually design, implement, test, and evaluate a complete multicore system capable of running simple parallel applications at the register-transfer level.

3. Objectives

This course is meant to be a capstone course in computer engineering that draws together concepts from across the ECE curriculum including digital logic design, computer organization, system-level software, computing, and engineering design. The course will prepare students for jobs in the computer engineering industry and can act as a springboard to more advanced material in graduate-level courses. This course can also provide a foundation for students interested in performance programming, compilers, and operating systems; and it can provide system-level context for students interested in emerging technologies and digital circuits. By the end of this course, students should be able to:

- **describe** computer architecture concepts and mechanisms related to the design of modern processors, memories, and networks and explain how these concepts and mechanisms interact.
- **apply** this understanding to new computer architecture design problems within the context of balancing application requirements against technology constraints; more specifically, quantitatively assess a design's execution time in cycles and qualitatively assess a design's cycle time, area, and energy.
- **evaluate** various design alternatives and make a compelling quantitative and/or qualitative argument for why one design is superior to the other approaches.
- **demonstrate** the ability to implement and verify designs of varying complexity at the register-transfer-level.
- **create** new designs at the register-transfer-level and the associated effective testing strategies.
- **write** concise yet comprehensive technical reports that describe designs implemented at the register-transfer-level, explain the testing strategy used to verify functionality, and evaluate the designs to determine the superior approach.

4. Prerequisites

This course is targeted towards senior-level undergraduate students, although it is also appropriate for advanced juniors, M.Eng., and first-year Ph.D. students. An introductory course on computing is required (CS 1110 or equivalent). A course in digital logic design and computer organization (ECE 2300 or equivalent) and a course in system-level programming (ECE 3140 or equivalent) are also required. CS 3410 is a suitable replacement for ECE 2300 and ECE 3140 for the purposes of satisfying the prerequisites. Students should feel comfortable working with a hardware description language such as Verilog, SystemVerilog, or VHDL and have a reasonable understanding of digital logic, assembly-level programming, storage systems, basic pipelining, and simple cache design.

M.Eng. and Ph.D. students coming from undergraduate institutions other than Cornell may want to spend additional time reviewing the secondary required textbook, *"Digital Design and Computer Architecture, 2nd edition"* by D. M. Harris and S. L. Harris (Morgan Kaufmann, 2012), to refresh their understanding of basic concepts. Students who have never used Python before may want to spend additional time reviewing the optional textbook titled *"Think Python: How to Think Like a Computer Scientist"* by A. B. Downey (Green Tea Press, 2014). Those students with less experience working with the Verilog hardware description language are strongly encouraged to read Chapter 4 in Harris and Harris on digital design with Verilog and/or to review the optional text *"Verilog HDL: A Guide to Digital Design and Synthesis, 2nd edition"* by S. Palnitkar (Prentice Hall, 2003). Students should also plan to attend the optional discussion sections, which will cover the basics of the Verilog language, how to use the course Verilog infrastructure, and some basic Verilog examples.

5. Topics

The course includes five parts: the first three parts cover the fundamentals of processor, memory, and network design, while the final two parts cover more advanced processor and memory design. In addition, the final lecture at the end of the course will present in detail an example architecture from industry to help illustrate the concepts discussed in class. A tentative list of topics for each part is included below. The exact topics covered in the course are subject to change based on student progress and interest.

- **Part 1: Fundamental Processors (4.5 lectures) –** instruction set architecture; single-cycle processors; hardwired vs. microcoded FSM processors; pipelined processors; resolving structural, data, control, and name hazards; analyzing processor performance
- **Part 2: Fundamental Memories (3.5 lectures) –** memory technology; direct-mapped vs. associative caches; write-through vs write-back caches; single-cycle, FSM, pipelined caches; analyzing memory performance
- **Part 3: Fundamental Networks (4 lectures) –** single-cycle global crossbars; arbitration; traffic patterns; torus and butterfly topologies; routing algorithms; channel and router microarchitecture; analyzing network performance
- **Part 4: Advanced Processors (9 lectures) –** superscalar execution, out-of-order execution, register renaming, memory disambiguation, branch prediction, speculative execution; multithreaded, VLIW, and SIMD processors
- **Part 5: Advanced Memories (3 lectures) –** advanced cache microarchitecture; memory protection, translation, and virtualization; memory synchronization, consistency, and coherence
- **Example Architecture (1 lecture) –** Intel Haswell

6. Required Materials

There are three materials that students are required to have access to for the course: the primary course textbook, the secondary course textbook, and the course packet. Note that if students are unsure about whether or not they will enroll in the class, they should be able to delay purchasing these items until the second or third week of class without significantly hindering their progress in the course. These materials are on reserve at Uris Library and/or available online to Cornell students.

- **Hennessy and Patterson Textbook –** The primary required textbook for the course is *"Computer Architecture: A Quantitative Approach, 5th ed.,"* by J. L. Hennessy and D. A. Patterson (Morgan Kaufmann, 2012). This is the classic text in the field, recently updated in 2012. The first chapter will be available on the course website for download. This textbook is on reserve in Uris library, and is also available as an e-book to any Cornell student with a valid NetID. A link to the e-book is on the public course website. Note that there is a limit to the number of simultaneous viewers, so please log out when finished with the e-book. Students can download up to 10-pages at a time, and print up to 60-pages at a time. There have been significant changes compared to earlier editions, but students may still be able to use an earlier edition augmented with the newer edition on reserve.
- **Harris and Harris Textbook –** The secondary required textbook for the course is *"Digital Design and Computer Architecture, 2nd ed.,"* by D. M. Harris and S. L. Harris (Morgan Kaufmann, 2012). This is the primary required textbook for ECE 2300. There will be some assigned reading from this book, but a student may or may not need to purchase the actual book depending on how comfortable a student is with the more basic material. M.Eng. and Ph.D. students coming from undergraduate institutions other than Cornell may want to purchase this book to solidify their understanding of digital logic design and basic computer architecture. The book also includes some useful background information on digital design using the Verilog hardware description language. This textbook is on reserve in Uris library, and is also available as an e-book to any Cornell student with a valid NetID. A link to the e-book is on the public course website. Note that there is a limit to the number of simultaneous viewers, so please log out when finished with the e-book. There have been some changes compared to the first edition, but students may still be able to use an earlier edition augmented with the newer edition on reserve.
- **Course Packet –** The course will use a packet of additional reading material on processors, memories, and networks that is meant to complement the course textbook. The information in the packet is either not in the textbook, or is presented in a useful alternative way. Two copies of the course packet are on reserve in Uris library along with the original books excerpted in the packet. Several of the excerpts in the book will also be available on the public course website for download, except for those which cannot be posted due to copyright restrictions.

7. Optional Materials

There are a few additional books that students may find useful for providing background on Python, Verilog, SystemVerilog, and the MIPS architecture.

- **Python Book –** *"Think Python: How to Think Like a Computer Scientist, Version 2.0.1"* by A. Downey (Green Tea Press, 2014) is an excellent introduction to Python especially well-suited for beginners to either the language or programming in general. This book is available on the course website for download.

- **Verilog Book –** *"Verilog HDL: A Guide to Digital Design and Synthesis, 2nd ed.,"* by S. Palnitkar (Prentice Hall, 2003) provides a good introduction to Verilog-2001 well suited for the beginner.
- **SystemVerilog Book –** *"SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling, 2nd ed.,"* by S. Sutherland, S. Davidmann, and P. Flake (Springer, 2006) provides a good introduction to the new features in SystemVerilog that can enable productive hardware design.
- **MIPS Book –** *"See MIPS Run Linux, 2nd ed.,"* by D. Sweetman (Morgan Kaufmann, 2006) is a great book about the MIPS instruction set and the hardware/software interface specifically with respect to running the Linux operating system.

8. Format and Procedures

This course includes a combination of lectures, short in-class quizzes, optional discussion sections, assigned readings, problem sets, laboratory assignments, and exams.

- **Lectures –** Lectures will be from 2:55pm to 4:10pm every Monday and Wednesday in 255 Olin Hall excluding the following academic holidays: Labor Day (9/7), Columbus Day (10/12), and Thanksgiving (11/25). We will start promptly at 2:55pm so please arrive on time. Students are expected to attend all lectures, be attentive during lecture, and participate in class discussion. Please turn off all cellular phones during class. Use of cellular phones and laptops during lecture is not allowed (see Section 11.C).
- **Quizzes –** There will be a short quiz at the beginning of some lectures. The quiz should take about five minutes, and will cover some of the key topics discussed in the previous lecture. Quizzes are not announced ahead of time. Students should prepare for a potential quiz by simply reviewing the material from the previous lecture before coming to class. Solutions to quizzes will be available online soon after the quiz is given for formative self-assessment.
- **Discussion Section –** The discussion section will be most Fridays from 2:30pm to 3:20pm in 203 Phillips Hall. Attendance at the weekly discussion sections is optional but strongly encouraged. These discussion sections will be relatively informal, with the primary focus being on facilitating student's ability to complete the lab assignments and on reviewing material from lecture using problem-based learning.
- **Readings –** Students are expected to complete all of the assigned reading according to the schedule on the course website, although there is some flexibility. Some students may prefer to complete the readings before the corresponding lecture, while others may prefer to complete the readings after the corresponding lecture. Either strategy is acceptable. The readings are contained within the course textbooks and the course packet.
- **Problem Sets –** The course will include four problem sets distributed throughout the semester to help you put the concepts learned in lecture and reading into practice. The problem sets are to be completed individually, although students are encouraged to study together and discuss information and concepts covered in lecture with other students (see Section 11.F for collaboration policy). Problem sets must be submitted in PDF format via the online CMS assignment submission system (see Section 12). You can use any program you want to compose your solutions, but you must convert the final document to PDF. Students are strongly encouraged to type their solutions; if necessary students can scan hand-written diagrams and combine these diagrams with typed solutions. If students absolutely must write up their entire solutions by hand, then they must scan their final version into a legible PDF. Illegible scans will not be

graded. A digital photograph will almost certainly not be legible. No other means of submission or electronic format will be accepted. Problem sets are due on Thursdays at 11:59pm (see Section 11.D for late assignment policy). Solutions to the problem sets will be available online soon after the problem set is due for formative self-assessment.

- **Lab Assignments –** The course will include five lab assignments that allow students to incrementally design, implement, test, and evaluate a complete multicore system with an integrated collection of processors, memories, and networks. Students are expected to work in a group of three students, although groups of two or four students may be allowed with explicit instructor permission in exceptional circumstances (see Section 11.F for collaboration policy). It is suggested that students form a group early on and keep the same group throughout the semester. Students can either form their own groups or ask the instructor to form a group for them. For each lab assignment, each student will take on a specific role: architect, design lead, or verification lead. Students must take on all three roles over the course of labs 2–4. Much more detail about the lab assignments is provided in the Lab Assignment Assessment Rubric posted on the public course website. Students will be using the ECE Computing Resources to complete the lab assignments, the lab code must be submitted via GitHub, and the lab report must be submitted in PDF format via the online CMS assignment submission system (see Section 12). No other means of submission or electronic format will be accepted. Lab assignments are due on Thursdays at 11:59pm (see Section 11.D for late assignment policy).
- **Midterm and Final Exams –** The course includes a midterm exam that covers Parts 1–3, and a final exam that covers the entire course with a focus on Parts 4–5. If students have a scheduling conflict with either the midterm exam or the final exam, they must let the instructor know as soon as possible, but no later than two weeks before the exam. Graded exams and the exam solutions are only available for review in 310/314 Rhodes Hall under the supervision of a course instructor. You may not remove your graded exam, nor may you remove the exam solutions from 310/314 Rhodes Hall.

9. Assignment and Exam Schedule

The current schedule is on the course website. All assignments are due on Thursdays at 11:59pm and should be submitted electronically via the online CMS assignment submission system except for the lab assignment code which should be submitted via GitHub (see Section 11.D for late assignment policy). Changes to this schedule will be posted as announcements via Piazza. Please note that there is either an assignment due or an exam almost every week. Students will need to manage their time carefully to succeed in this course.

Thu	Sep	10	Lab 1 – Iterative Integer Multiplier
Thu	Sep	24	Problem Set 1
Thu	Oct	1	Lab 2 – Pipelined Processor
Thu	Oct	8	Problem Set 2
Thu	Oct	15	Midterm Exam from 7:30–10:30pm in B14 & 110 Hollister Hall (covers Parts 1–3)
Thu	Oct	29	Lab 3 – Blocking Cache
Thu	Nov	5	Problem Set 3
Thu	Nov	12	Lab 4 – Ring Network
Thu	Nov	19	Problem Set 4
Thu	Dec	3	Lab 5 – Multicore System
	Dec		Final Exam (covers Parts 1–5)

10. Grading Scheme

Each part or criteria of every quiz, assignment, and exam is graded on a four-point scale. A score of 4.5 is an A+, 4 roughly corresponds to an A, 3 roughly corresponds to a B, 2 roughly corresponds to a C, and so on. A score of 4.0 usually indicates that the submitted work demonstrates no misunderstanding (there may be small mistakes, but these mistakes do not indicate a misunderstanding). A score of 3.0 usually indicates that the submitted work demonstrates more understanding than misunderstanding. A score of 2.0 usually indicates that the submitted work demonstrates a balanced amount of understanding vs. misunderstanding. A score of 1.0 usually indicates that the submitted work demonstrates more misunderstanding than understanding. A score of 4.5 is reserved for when the submitted work is perfect with absolutely no mistakes or is exceptional in some other way.

Total scores are a weighted average of the scores for each part or criteria. Parts or criteria are usually structured to assess a student's understanding according to four kinds of knowledge: basic recall of previously seen concepts, applying concepts in new situations, qualitatively and quantitatively evaluating design alternatives, and creatively implementing new designs; these are ordered in increasing sophistication and thus increasing weight. In almost all cases, scores are awarded for demonstrating understanding and not for effort. Detailed rubrics for all quizzes, problem sets, and exams are provided once the assignment has been graded to enable students to easily see how the score was awarded. For lab assignments, a detailed Lab Assignment Assessment Rubric is available on the public course webpage.

Note that only a subset of the problems on each problem set may be graded for a score. Which problems will be graded for a score will not be announced ahead of time; students are expected to complete all problems. Those problems not graded for a score will still be graded for effort, and incomplete submissions will be penalized. As mentioned above, solutions to all problems are released soon after the problem set is due; this will allow students to compare their submission to the correct solutions for formative self-assessment. Students are always encouraged to review their solutions with the instructor or TAs during office hours.

The final grade is calculated using a weighted average of all assignments. Each problem set and lab assignment is weighted equally. All quiz grades are averaged to form a single total. Students can drop their lowest three quiz scores. At the instructor's discretion, additional pseudo-quiz grades may be used to encourage participation, completing student evaluations, etc. The weighting for the various assignments is shown below.

Quizzes	5%	(students can drop lowest three scores)
Problem Sets	15%	(all problem sets weighted equally)
Lab Assignments	30%	(all lab assignments weighted equally)
Midterm Exam	20%	
Final Exam	30%	

Note that the midterm and final exam account for half of a student's final grade. The exams in this course are very challenging. Successful students begin preparing for the exams far in advance by carefully reviewing the assigned readings, independently developing study problems, and participating in critical study groups.

To pass the course, a student must at a bare minimum satisfy the following requirements: (1) submit two out of the four problem sets; (2) submit three out of the five lab assignments; (3) take the midterm exam; and (4) take the final exam. **If a student does not satisfy these criteria then that student may fail the course regardless of the student's numerical grade.**

11. Policies

This section outlines various policies concerning auditors, usage of cellular phones and laptops in lecture, turning in assignments late, regrading assignments, collaboration, and accommodations for students with disabilities.

11.A Auditor Policy

Casual listeners that attend lecture but do not enroll as auditors are not allowed; you must enroll officially as an auditor. Auditors are allowed to enroll in the course as long as there is sufficient capacity in the lecture room. The requirements for auditors are: (1) attend most of the lectures; (2) take the short in-class quizzes; and (3) perform reasonably well on these quizzes. If you do not plan on attending the lectures for the entire semester, then please do not audit the course. Please note that students are not allowed to audit the course and then take it for credit in a later year unless there is some kind of truly exceptional circumstance.

11.B Course Re-Enrollment Policy

Students are not allowed to enroll for credit for a significant fraction of the course, drop or switch to auditor status, and then re-enroll for credit in a later year. A "significant fraction of the course" means after the second problem set is due; by this time the student will have: attended eight lectures, completed two lab assignments and two problem sets, and completed several short in-class quizzes. The student should have plenty of experience to decide whether or not they should drop and take the course in a later year. It is not fair for students to have access to assignment solutions and possibly even take the midterm before deciding to drop the course and take it again in a later year; this would essentially enable students to take the course twice to improve their grade.

11.C Cellular Phones and Laptops in Lecture Policy

Students are prohibited from using cellular phones and laptops in lecture unless they receive explicit permission from the instructor. It is not practical to take notes with a laptop for this course. Students will need to write on the handouts, quickly draw pipeline diagrams, and sketch microarchitectural block diagrams during lecture. The distraction caused by a few students using (or misusing) laptops during lecture far outweighs any benefit. Tablets are allowed as long as they are kept flat and used exclusively for note taking. If you feel that you have a strong case for using a laptop during lecture then please speak with the instructor.

11.D Late Assignment Policy

Problem sets must be submitted electronically in PDF format, and lab assignments must be submitted electronically as a tar/gzip file (as explained in the lab handout). **No other formats will be accepted!** Problem sets and lab assignments must be submitted by 11:59pm on the due date. No late submissions will be accepted and no extensions will be granted except for a family or medical emergency. We will be using the online CMS assignment submission system. You can continue to resubmit your files as many times as you would like up until the deadline, so please feel free to upload early and often. **If you submit an assignment even one minute past the deadline, then the assignment will be marked as late.**

As an exception to this rule, each student has a set of slip days that may be used when submitting problem sets and lab assignments throughout the semester. Each slip-day provides an automatic 24-hour extension. You may use up to two slip-days on any single assignment, meaning that the

maximum automatic extension is 48 hours. For lab assignments, we use the majority to determine how many slip days. So if after some changes in lab groups, two students have three slip days and one student has one slip day then the group essentially now has three slip days available. If one student has three slip days, one student has two slip days, and one student has one slip day, then the group essentially now has two slip days. **Students have three slip days exclusively for use on problem sets and three slip days exclusively for use on lab assignments.** These slip days are not interchangeable. To use a slip day, simply submit your assignment late; CMS will allow assignments to be uploaded up to two days late. You are responsible for keeping track of how many slip days you have remaining. If you accidentally submit an assignment late without the proper number of slip days remaining then although the system will allow the upload we will not grade the assignment (or we will grade the latest upload before the due date). The purpose of the slip-day system is to give you the freedom to more effectively manage your time. The due dates for the course are available at the beginning of the semester, so please plan ahead so you can handle weeks with many other deadlines.

11.E Regrade Policy

Addition errors in the total score are always applicable for regrades. Regrades concerning the actual solution should be rare and are only permitted when there is a significant error. Please only make regrade requests when the case is strong and a significant number of points are at stake. Regrade requests should be submitted online through the CMS assignment submission system within one week of when an assignment is returned to the student. You must provide a justification for the regrade request.

11.F Collaboration Policy

The work you submit in this course is expected to be the result of your individual effort only, or in the case of lab assignments, the result of you and your group's effort only. Your work should accurately demonstrate your understanding of the material. The use of a computer in no way modifies the standards of academic integrity expected under the University Code.

You are encouraged to study together and to discuss information and concepts covered in lecture with other students. You can give "consulting" help to or receive "consulting" help from other students. Students can also freely discuss basic computing skills or the course infrastructure. **However, this permissible cooperation should never involve one student (or lab group) having possession of or observing in detail a copy of all or part of work done by someone else, in the form of an email, an email attachment file, a flash drive, a hard copy, or on a computer screen.** Students are not allowed to seek consulting help from online forums outside of Cornell University. Students are not allowed to use online solutions (e.g., from Course Hero) from previous offerings of this course. Students are encouraged to seek consulting help from their peers and from the course staff via office hours and the online Piazza discussion forums. **If a student receives consulting help from anyone outside of the course staff, then the student must acknowledge this help on the submitted assignment.**

During examinations, you must do your own work. Talking or discussion is not permitted during the examinations, nor may you compare papers, copy from others, or collaborate in any way. Students must not discuss an exam's contents with other students who have not taken the exam. If prior to taking it, you are inadvertently exposed to material in an exam (by whatever means) you must immediately inform the instructor or a TA.

Should a violation of the code of academic integrity occur, then a primary hearing will be held. See `http://www.theuniversityfaculty.cornell.edu/AcadInteg` for more information about academic integrity proceedings.

Examples of acceptable collaboration:

- Bob is struggling on a problem set about processor pipelining, so he seeks consulting help from Alice, a fellow student in the course. Alice goes through various examples from the lecture and reading materials to help Bob understand the concepts, and they sketch a few pipeline diagrams related to the problem solution together on a whiteboard. Bob and Alice work independently to flesh out the details of the problem solution and they each write up their work independently. Bob acknowledges the help he received from Alice on his submission.

- Bob, Ben, and Beth are struggling to complete a lab assignment which requires implementing a direct-mapped cache. They talk with Alice, Amy, and Adam and learn that both groups are really struggling. So the six students get together for a brainstorming session. They review the lecture and reading materials and then sketch on a whiteboard some ideas on how to implement a direct-mapped cache. They might also sketch out some code snippets to try and understand the best way to describe some of the hardware. Then each group independently writes the code for the assignment and includes an acknowledgment of the help they received from the other group. At no time do the groups actually share code.

- Bob, Ben, and Beth are having difficulty figuring out difficult test cases for their pipelined processor. They make a post on Piazza to see if anyone has some general ideas for tricky corner cases. Alice, Amy, and Adam figured out an interesting test case that ensures their pipelined processor correctly forwards the address to a store instruction, so Alice, Amy, and Adam post a qualitative description of this test case. Bob, Ben, and Beth independently write the code for this test case and then include an acknowledgment of the help they received from the other group. At no time do the groups actually share test code.

Examples of unacceptable collaboration:

- Bob is struggling on a problem set about processor pipelining, so he seeks consulting help from Alice, a fellow student in the course. *Alice shows Bob her completed problem set solutions and walks him through the various steps required to solve the problem.* Bob takes some notes during their discussion, and then independently writes up his solutions. Bob acknowledges the help he received from Alice on his submission, but it doesn't matter since Alice explicitly shared her solutions with Bob.

- Bob, Ben, and Beth are struggling to complete a lab assignment which requires implementing a direct-mapped cache. They talk with Alice, Amy, and Adam and learn that both groups are really struggling. So the six students get together for a joint coding session. Each student works on one module in the cache, then they combine the modules together to create the final working direct-mapped cache. *The six students share and copy each others code often in order to finish the assignment.* Each group submits the final code independently. Each group acknowledges the help it received from the other group, but it doesn't matter since they explicitly shared code.

- Bob, Ben, and Beth are having difficulty figuring out difficult test cases for their pipelined processor. They make a post on Piazza to see if anyone has some general ideas for tricky corner cases. Alice, Amy, and Adam figured out an interesting test case that ensures their pipelined processor correctly forwards the address to a store instruction, so *Alice, Amy, and Adam send their test code to Bob, Ben, and Beth via email.* Bob, Ben, and Beth modify this test code and then include

it in their submission. Bob, Ben, and Beth include an acknowledgment of the help they received from the other group, but it doesn't matter since they explicitly shared code.

Notice that **the key is that students should not share the actual solutions or code with each other.** Consulting with your fellow students is fine and is an important part of succeeding in this course. If the vehicle for consulting is a whiteboard (and you avoid writing the actual solution on the whiteboard) then you should be fine.

11.G Accommodations for Students with Disabilities

In compliance with the Cornell University policy and equal access laws, the instructor is available to discuss appropriate academic accommodations that may be required for students with disabilities. Requests for academic accommodations are to be made during the first three weeks of the semester, except for unusual circumstances, so arrangements can be made. Students are encouraged to register with Student Disability Services to verify their eligibility for appropriate accommodations.

12. Online and Computing Resources

We will be making use of a variety of online websites and computing resources.

- **Public Course Website –** `http://www.csl.cornell.edu/courses/ece4750`
  This is the main public course website which has the course details, updated schedule, reading assignments, and most handouts.
- **Piazza Discussion Forums –** `http://www.csl.cornell.edu/courses/ece4750/piazza`
  Piazza is an online question-and-answer platform. We will be using Piazza for all announcements and discussion on course content, problem sets, and lab assignments. We will enroll students that sign up for the course in Piazza. The course staff is notified whenever anyone posts on the forum and will respond quickly. Using the forum allows other students to contribute to the discussion and to see the answers. Use common sense when posting questions such that you do not reveal solutions. Please prefer posting to Piazza as opposed to directly emailing the course staff unless you need to discuss a personal issue.
- **CMS –** `http://www.csl.cornell.edu/courses/ece4750/cms`
  CMS is an online assignment management system developed by the Cornell Computer Science department. Students will use CMS for all submitting all assignments. The only exception is for the lab assignment code which will be submitted via GitHub. We will enroll students that sign up for the course in CMS. We will also be posting restricted materials (e.g., problem sets, quizzes, solutions) on the CMS site, and we will use CMS to distribute grades. We will not be using CMS to post announcements.
- **ECE Computing Resources –** `http://www.csl.cornell.edu/courses/ece4750/account`
  The ECE department has a cluster of Linux-based workstations and servers which we will be using for the lab assignments. You can access the ECE computing resources by using the ECE Linux Computing Lab in 314 Phillips Hall, you can use the CIT Windows Computing Lab in 318 Phillips Hall, or you can log into the `amdpool` servers remotely from your own personal workstation. All laboratory assignments will need to be completed using your ECE account.
- **GitHub –** `http://www.csl.cornell.edu/courses/ece4750/github`
  GitHub is an online Git repository hosting service. We will be using GitHub to distribute lab assignment harnesses and as a mechanism for student collaboration on the lab assignments.

Students will also use GitHub for submitting the code-portion of their lab assignments. Students are expected to become familiar with the Git version control system.

- **TravisCI –** `http://www.csl.cornell.edu/courses/ece4750/travisci`
  TravisCI is an online continuous integration service that is tightly coupled to GitHub. TravisCI will automatically run all tests for a students' lab assignment every time the students push their code to GitHub. We will be using the results reported by TravisCI to evaluate the code functionality of the lab assignments.