Course Syllabus

ENEE 350H: Computer Organization, Fall 2009
Prof. Bruce Jacob

Basic Information

Time & Place

Lecture: TuTh 9:30–10:45 pm, EGR-1108
Discussion Section: Mon 11:00–11:50am, CHE-2136

Professor

Bruce L. Jacob: AVW-1325, blj@ece.umd.edu
Office hours: *open-door policy* (for now …)

Teaching Assistant

Yufu Zhang, yufuzh@gmail.com/yufuzh@umd.edu

Class Home Page

http://www.ece.umd.edu/courses/enee350h/

Class Email List

enee350h-0101-fall09@coursemail.umd.edu

Class Schedule

This is a weekly schedule of my hours, including class time and scheduled office hours, but also including other things that make me unavailable. It is subject to change.

<table>
<tr><th></th><th>MON</th><th>TUE</th><th>WED</th><th>THU</th><th>FRI</th></tr>
<tr><td>9–9:30</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>9:30–10</td><td></td><td rowspan="3">ENEE 350H
Lecture EGR-1108</td><td></td><td rowspan="3">ENEE 350H
Lecture EGR-1108</td><td></td></tr>
<tr><td>10–10:30</td><td></td><td></td><td></td></tr>
<tr><td>10:30–11</td><td></td><td></td><td></td></tr>
<tr><td>11–1:30</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>11:30–12</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>12–12:30</td><td></td><td rowspan="4">ENES 100
Lecture JMP-1116</td><td></td><td rowspan="4">ENES 100
Lecture JMP-1116</td><td rowspan="5">Meetings with graduate students</td></tr>
<tr><td>12:30–1</td><td></td><td></td></tr>
<tr><td>1–1:30</td><td></td><td></td></tr>
<tr><td>1:30–2</td><td></td><td></td></tr>
<tr><td>2–2:30</td><td></td><td></td><td></td><td></td></tr>
<tr><td>2:30–3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3–3:30</td><td rowspan="4">ENEE 359R
Lecture
AVW-2446</td><td></td><td></td><td></td><td></td></tr>
<tr><td>3:30–4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4–4:30</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4:30–5</td><td></td><td></td><td></td><td></td></tr>
</table>

Course Overview

This course is intended to give you a basic understanding of how computers execute programs. Understanding computers means understanding the hardware/software process of how you and the computer work together to have the computer carry out a concept. In your introductory computer courses that used or taught programming (e.g. ENEE 114), you learned how to express a concept in terms of a high-level programming language such as C. ENEE 244 taught you the basics building blocks of hardware (flip-flops, gates, etc.). ENEE 350 is the course where these two approaches meet. You will see in this course how a low-level language is executed by the hardware, and you will see how to put together basic hardware building blocks to form the functional units of a computer.

To achieve these goals, you will partially design and "build" simple computers at various levels of detail. Building in this course will not mean connecting chips and gates (we assume you can do that from ENEE 244). Rather, you will describe the hardware in diagrams, finite-state machines, and hardware simulators written in C.

Prerequisites

Students must have taken ENEE 244, or have equivalent knowledge of digital logic design. Students should also have taken ENEE 114, though it is not a requirement. You should understand digital design concepts such as multiplexors, gates, boolean algebra, finite-state machines, and flip-flops. You should also understand and be reasonably fluent in programming in C, e.g. using arrays, structures, functions, and pointers.

You may not yet be familiar with programming in C on Unix. I advise that you learn it quickly—your TA can help during office hours.

Course Material

The required text for the course:

> *Computer Organization and Design: The Hardware/Software Interface, 3rd Ed.*, by Patterson and Hennessy.

You will be responsible for the material in this book. READ IT. It is a very well-written book and contains **lots** of real-world examples and advice. READ IT. The authors know what they are talking about; they have **both** built well-known, highly influential computer systems. READ IT.

Class Projects

Four projects will be assigned during the term, each of which will require a substantial time commitment on your part. You may find the work load in this course to be heavy.

The most common reason for not doing well on projects is not starting them early enough. You will be given plenty of time to complete each project. However, if you wait until the last minute to start, you may not be able to finish. Plan to do some work on a project every day. Also plan to have it finished about 2 days ahead of the due date—many unexpected problems arise during programming, especially in the debugging phase. The computing sites can become quite crowded as deadlines approach, making it difficult to get a computer. **Plan** for these things to happen. Your lack of starting early is not an excuse for turning in your project late, even if some unfortunate situations arise such as having your computer crash.

There are many sources of help on which you can draw. Simple questions can be submitted to the TAs, professor, and fellow classmates via email (**use the email list given on page 1**). These will typically be answered within the day, often more quickly during working hours. Keep in mind, however, that many types of questions cannot be answered without seeing your project. If you have detailed questions, your best option is to speak to the TA or professor in person during office hours. Bring along a listing of your project, and the output from a run if available. Students are also encouraged to help one another. *One of the best ways for you to make sure that you understand a concept is to explain it to someone else.* Keep in mind, however, that you should not expect anyone else to do any part of your project for you. The project that you turn in must be your own.

Many computing sites have consultants who are available to help you at the site. They are fine sources of help with questions regarding the computers and installed software (such as Unix, email, and the C compiler). However, they are not likely to be able to help you with questions about computer programming, the C language, or specific errors in your program.

Turning in Projects

Projects are due at 5:00 pm on the due date. We will allow a grace period and accept projects until 11:59 pm. Sometimes unexpected events make it difficult to get a project in on time. For this reason, each person will have a total of 3 free late days to be used for projects throughout the semester. **These late days should only be used to deal with unexpected problems such as computer crashes, illness, or submission problems.** They should not be used simply to start later on a project or because you are having difficulty completing the project. Projects received after the due date (assuming that you have no late days left) will receive a zero, even if it is just one second late. I advise you to save at least one or two late days for the last project. Weekend days are counted in exactly the same way as weekdays (e.g. if the project deadline is Friday and you turn it in Sunday, that's two days late). You will be submitting your projects electronically by using the make facility—the same facility you will use to build your projects. I will provide you with a Makefile for each project.

Extensions

Extension requests (other than the use of free late days) will be considered only if you ask the professor *before* the original due date. Extensions will only be granted for medical or personal **emergencies**. Be prepared to substantiate any extension request you make with written proof, for example a written note from your doctor. Extensions are not granted for reasons such as: the printer went down, you erased all your files, you lost your program printout, the terminal room was crowded and you couldn't get a terminal, you had other course work or job commitments which interfered, etc. You can avoid all these problems by starting the projects early and keeping backup files. If you are having trouble understanding the material or designing a program, please come to office hours for help right away.

Project Grading

The projects will be graded primarily for correctness (doing all the required tasks, simulating at the correct hardware level, and giving correct results). All grading questions should first be discussed with your TA. If you cannot resolve a problem with the TA, bring the project to the instructor.

Homeworks

Homeworks prepare you for the exams. There will be two types: **graded** and **non-graded**. Non-graded homeworks will largely be problems from the textbook that will be covered in the weekly

discussion sections led by the TA. Graded homeworks will contain questions similar to those you will find on the exams. They will be collected at the beginning of lecture and graded on a check/check-plus/check-minus scale. **There will be no late days for homeworks, and homeworks will not be accepted after the beginning of lecture.**

Doing Your Own Work

There are two types of assignments in this class; not surprisingly, we have several different attitudes toward collaboration. In general, all work in this course is to be done on your own. However, at the same time, we want students to help each other learn the course material. As in most courses, there is a boundary separating these two situations. You may give or receive help on any of the topics covered in lecture or discussion and on the specifics of C syntax. You are allowed to consult with other students in the current class during the conceptualization of a project. You are not allowed to use the work or specific ideas of other students.

If you have any questions as to what constitutes unacceptable collaboration, please talk to the instructor right away. You are expected to exercise reasonable precautions in protecting your own work; for instance, do not leave a copy of your assignment in a publicly accessible directory, and take care when discarding hardcopy.

Collaboration on Projects

All work on projects is to be your own. Violation will result in a zero on the project or exam in question and initiation of the formal procedures of the Student Honor Council. We will be using an automated program to correlate projects.

You may discuss tactics and techniques for solving programming problems, but you are not to stray too far into the details of the solution, else the automated checker might find similarities between different projects. In general, all written work, whether in scrap or final form, must be generated by you working alone. You are not allowed to work out the programming details of the problems with anyone or to collaborate to the extent that your programs are identifiably similar.

Collaboration on Homeworks

In contrast, we highly encourage cooperation on homework assignments, as this is a very effective way for you to learn the material, provided that you think through all the problems. However, if you simply copy someone else's homework, you'll do poorly on the exams, which count five times as much as the homeworks—probably not a good trade-off.

Exams

You are expected to take both the midterm and final exams at the scheduled times. Unless a (documented) medical or personal emergency is involved in your missing an exam, you will receive a zero for that exam. If you anticipate conflicts with the exam time, you must come talk to the instructor about it at least **1 month** before the exam date. The exam dates are given at the beginning of the term so that you can avoid scheduling job interviews or other commitments on exam days. Outside commitments are not considered a valid reason for missing an exam. **Exams will be closed book, closed notes.**

Grading Policy

Final grades will be based on the total of points earned on the projects, homeworks, and exams. The tentative point breakdown is as follows:

- Projects: 40%
- Homeworks: 10%
- Midterm Exam: 20% (Thursday, 28, in class)
- Final Exam: 30% (Friday, May 15 at 8:00 a.m.)

The grade of "Incomplete" will generally not be given. According to university policy, doing poorly in a course is not a valid reason for an incomplete. If you are having problems in the course, your best bet is to come talk to the instructor as soon as you are aware of it.

Computers

You may use any departmental Sun workstation that runs *gcc*. I will distribute a Unix handout that contains more details about compiling and submitting projects.

Tentative Lecture Schedule

Week of	Subject	Readings	Projects
Aug 31	Intro to course	Chap. 1, 4	**P1 out**
Sep 7	Machine-level instructions	Chap. 2, A	
Sep 14	Machine-level instructions, machine arithmetic	Chap. 2, A, 3	**P1 due**
Sep 21	Machine arithmetic, processor implementation	Chap. 3, 5, B, C	**P2 out**
Sep 28	Processor implementation	Chap. 5, B, C	
Oct 5	Processor implementation	Chap. 5, B, C	
Oct 12	Performance	Chap. 4	**P2 due**
Oct 19	**Review & Midterm** (Thursday, October 22, in class)		**P3 out**
Oct 26	Pipelining	Chap. 6	
Nov 2	Pipelining	Chap. 6	
Nov 9	Memory hierarchy	Chap. 7	**P3 due**
Nov 16	Memory hierarchy, Operating systems	Chap. 7 & readings	**P4 out**
Nov 23	Operating systems, Storage architectures	Chap. 8 & readings	
Nov 30	Advanced stuff	Chap. 9 & Readings	
Dec 7	**Final Review**		**P4 due**
Exams	**Final Exam** (Tuesday, December 15, 8:00 a.m.)		

Special Needs

If you have a documented disability that requires special needs, please see me as soon as possible, and certainly no later than the third week of classes.