EE 345 / 445 Computer Organization Fall, 2013

Texts: David A Patterson and John L. Hennessy, *Computer Organization & Design The Hardware / Software Interface*, 4th ed., Morgan Kaufmann
Software: Altera Quartus II (download to use with FPGA boards)
Scheduled class times: MW 9-10:45 AM* Location SLC224
Instructor: John B. Gilmer Jr. Office hours: TBD Office:SLC220 Phone: x4885
Prerequisite: EE241 Digital Design or equivalent

Background:

This course deals with the fundamentals of how computers work. The student has already gained an understanding of how semiconductor devices: transistors, diodes and other components, can be assembled into circuits that perform basic logic functions such as AND, OR, or simple memory cells. Furthermore, we have digital design techniques for assembling gates and flipflops into Medium Scale Integration devices such as adders, registers, counters, and multiplexers. These circuits can be used in larger ones that perform functions as complicated as controlling a traffic signal, playing tic-tac-toe, or a simple calculator.

All of these machines discussed so far are wired to perform one particular function. The digital computer has proved to be such a useful device because it is not limited to performing just a single function. The same assemblage of wires and chips can do different things, because the function to be performed is encoded in a stored "program" rather in the connections of the wires. In addition, it has proved possible to construct programs that are much more complicated, and perform much more elaborate functions, than would be practical if the same function was to be built entirely with hardware. Even when we want a controller to do only one thing, it is often easier to take a standardized piece of hardware, for example a single chip microcontroller, program it to do our function, then embed it into the machine to be controlled, rather than build custom hardware.

The concept of the stored program is central to the conventional modern digital computer. The hardware of a computer consists of a "data path" which has circuitry consisting of busses, registers, and arithmetic units that can move data around and perform calculations. You can think of it as being like a calculator. The program consists of instructions, which are fetched one by one, which each direct the data path, through a control unit, to perform some specific operation. These instructions are stored in the machine's memory array. You can think of the control unit, under direction of the program, as pushing the buttons on the calculator. In this course, we be focusing on how the data path is organized, how the control unit can be built, options and possibilities for formatting instructions, and ways to build computers that go faster.

Today, most programs for digital computers are written in a "higher level language" such as C rather than directly in the machine instructions that the computer actually executes. Translation is performed by "compilers". The student should already be familiar with the nature of stored programs: that lines of code are executed one at a time, and "variables" stored in the memory and program flow are affected by calculations performed. Such programs are compiled into machine language form before executing. We will look at issues concerning how instructions need to be sequenced in order to perform well on modern "pipelined" processors, which execute their instructions in assembly-line fashion. A prior knowledge of machine language is not necessary. Machine language is covered in this course, but we are more interested in understanding it, and how it represents what the hardware can do, rather than becoming proficient programmers. (In a Machine Language course, or some other course focused on how the operating system interacts with the

hardware, there would be more attention to system aspects, e.g. how large programs are constructed, libraries, interfaces to networks, and such. Here, we are more interested in how the instructions execute on the hardware of the machine.)

In summary, this course is a bridge between an understanding of the basic digital circuits and principles, and the technology of software. The student should gain an appreciation of the technologies and principles that make possible the execution of simple programs. This is where the magic happens: a piece of electronic machinery actually does the fundamental operations that make possible the elaborate networks, simulations, and graphics that we take for granted every day.

Structure of the Course:

The text for this course contains more material than we can cover in one course, and we will necessarily be selective in what we will be able to cover. In the initial week of the course we will start with some introductory material based on a text by Malvino, called a "Simple-As-Possible #1" (SAP-1) computer, and its elaboration the SAP-2. This was first shown in EE241 to illustrate where we were going. We are also going to study the HCS08 machine language and CPU, and consider how a CPU of this sort would be built using the principles learned in the course. Finally, the book presents the "MIPS" architecture, a modern RISC type 32 bit machine, which makes a very interesting comparison to the SAP types, the HCS08, and the 68000/Coldfire (which we will also look at).

The use of a project is important to this course. It is sometimes difficult to fully grasp the principles we will be studying, if they are left as abstract exercises. Building a simple computer, making the design choices and tradeoffs necessary to do so, and discovering and correcting faults is a valuable learning experience. For the project, we will build computers "inside" an FPGA (Field programmable Gate Array). An FPGA is related to the GAL's that you may have used in EE241, having eight to twelve "macrocells" that could be programmed to perform simple functions. An FPGA is similar in principle, but has on the order of millions of macrocells, and wiring resources to connect them in a variety of ways. You can think of the FPGA as a programmable breadboard populated with gates for which you just need to specify the functions and connections. The programming is done using the Integrated Development Environment "Quartus II" for the Altera devices we are using. The user functional specification can include schematics, functional descriptions in the Verilog or VHDL hardware description language, or a combination of both. Learning about FPGA's is a very helpful side benefit of this course. (We won't be doing any hardware construction with wires or parts or breadboards. Everything we need is already on t6he FPGA "demo" boards. Think of them as being like the small microcontroller boards used for Mechatronics, but a lot bigger, more capable, and using an FPGA instead of a microcontroller.

The projects will be to build a computer in your FPGA that will execute a relatively simple program. We will work in teams of two (or possibly three). It will be your own custom design, implementing your own set of instruction formats and operation codes. I plan to have a base design available to use as a departure point.

Notice that the room scheduled for the course is SLC224. This fits in with the project focus. I expect that some of our sessions may be project work, with questions coming up and the problems and issues presented to the class as a whole. So, you will want and need to do your project work outside of normal class times, then bring current issues and problems to class, and we can address them right then and there. This will help the whole class understand each of the projects, and the different issues faced. We are going to be pretty informal about it. I do expect people to be in class though! (SLC224 has been recently renovated, and should have everything needed for the projects.)

Week of:	Topics covered	Reading	Tests
1 Aug 31	Overview, Number and Instructions, SAP intro.	Chapter 1, notes, 4.1-3	
2 Sept 2*	Elaborations on "Simple-as-possible" computer	Notes	
3 Sept 7	Microprocessors, the HCS08 and 68000, performance	Notes, Chapter 2	
4 Sept 14	Measurement principles, Instruction basics	Chapter 3.1-4	
5 Sept 21	Introduction to FPGA's, FPGA based design	Quartus material	
6 Sept 28	Instructions: program control	Chapter 3 rest, notes (68000)	
7 Oct 5	The Data Path: basic operations and components	Chapter 4.4,4.4	test#1
8 Oct 12	The Data Path: multiplication, division, float. pt.	Chapter 4,rest	
9 Oct 19	The control unit: Control and Microprogramming	Chapter 5.1-3	
10 Oct 26	Control and Microprogramming (continued)	Chapter 5.4-5.5	
11 Nov 2	Advanced concepts: pipelined execution	Chapter 6	test#2
12 Nov 9	Cache Memory, Virtual memory (projects)	Chapter 7	
13 Nov 16	Superscalar, shared memory, parallel processing (projects)		
14 Nov 23*	Array processing, VLIW, other advanced concepts (projects)		
15 Nov 30	Project related stuff: critique, testing, analysis, demonstration		
16 Exam week	Exam	(comprehensive)	Exam

* indicates a "short" week with only one class meeting.

Grading:

I intend to grade this course a bit differently than usual. Because of the project focus, I will make the project worth 50% of the course grade for those who have done a good job on the project (that is, if it will give you a better grade). All others will be graded by the "nominal" allocations given below. If the project expands to 50%, the allocation to tests and exam shrink in proportion to make up 46% of the grade. I'll calculate everybody's grade both ways, and give whichever works out for the best.

Tests will cover all material through the previous week. Tests will generally be on Wednesday of the week listed, unless an announcement is made setting the date differently. (I'm open to moving them; let me know if that's needed.) Tests are open book. They are hard. You will not have time to open your book very much, and still complete the test. Be well prepared. The two tests are given in the first half of the course since the projects will be more the focus of attention later. The exam will be comprehensive, but will have a large emphasis on the later topics.

There will be several homework assignments. Some will involve writing and executing short programs. (I may include possibly compiling and running them on the HCS08 processors; we'll see if that makes sense. I have some boards similar to thiose from Mechatronics.) Other assignments will require the use of Quartus II, in some cases doing something with the FPGA boards. The intent is to give you some practice on meaningful problems, and help prepare for the projects. Homework solutions will be reviewed at the beginning of the class when the assignments are due. These assignments will be ungraded, but I will take them up to see how students have done.

"Nominal" Grading Allocation:

2 tests at 20% each :	40%	project grades	21%
class participation (do HW? attend?...)	4%	final examination:	35%

All material will be graded on a basis of 0-100, with most graded material allowing for grades higher than 100 with bonus questions (usually up to 10% extra) considered. On tests and the

examinations some questions may be "compensated" if large numbers of students miss them (indicating possibly a badly posed question or inadequate coverage of the topic in class). On such questions, some proportion of the "lost" credit will be returned. This is the only form of "curving" of grades in the course. All written work is expected to be neat and well presented. A penalty of up to 10% will be assessed for poor presentation on any written work, and up to 30% on projects. I will not collect late homework. (Notice: I have no plans for "pop" quizzes. If there are any, they will just count as part of "participation".)

The grades from all work will be weighted as given in the above table, totaled, and converted into the Wilkes 4.0 scale grading system using the following conversion:

93+:	4.0	83-87:	3.0	70-76:	2.0	60-64:	1.0
88-92:	3.5	77-82:	2.5	65-69:	1.5	below 60:	0.0

All graded material handed in is to be the student's own work. Since the homeworks are not graded, you may receive help on these or even work with another student. However, if you do this, please indicate the degree of your own involvement. Don't just submit a xerox copy of another student's work. Likewise, don't just be a manual Xerox copier. The degree to which students participate in doing homeworks will be subjectively judged and may influence the final grade by up to a point in either direction in borderline cases, as well as affecting the subjective "class participation" part of the grade. The intent here is to allow any degree of cooperation and help on the homework.

Notes:

A notebook of class notes may be kept in the library on reserve, or perhaps notes will be provided as needed. This will include some of the lecture materials used, worked homework assignments, and test solutions. You are not obligated to copy any of the library reserve materials; it is merely meant to be helpful. Any material that is really needed will be distributed in the form of handouts in class (e.g. data on devices not in your references or texts, etc.). There will be lots of that, since you will need material on the SAP, the 68000 and some other topics.

Final Cautions:

There are always the possibilities of problems and issues in the use of simulation software and design software. We have Quartus II and FPGA use in the course, and occasionally we have run into licensing issues or technical challenges in the past. Some of the details have not yet been worked out as of this writing. It is possible that the schedule and even some of the intended features of the projects will need to change.

I'm looking forward to this course. I hope you are too. It should be interesting.