CS/EE333 – Computer Architecture

Chris Milner (Fall 2003)

Course Objectives:

1. Understand the classification of computers (accumulator machines, stack machines and general purpose register machines), instruction types (arithmetic, data movement and control), instruction formats (0,1,2, etc.-address machines) and addressing modes.
2. Understand formal notations for describing processors.
3. Evaluate the design and performance trade-offs between Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC) .
4. Develop an understanding of processor design (the design process, data path implementation, control unit implementation, 1- 2- and 3-bus processor designs and machine exceptions, pipelining and instruction-level parallelism)
5. Develop an understanding of computer arithmetic and arithmetic units
6. Develop an understanding of the memory hierarchy, cache memory and virtual memory.
7. Gain practical experience in programming with assembly language.
8. Understand program vulnerabilities arising from computer architecture decisions.

Mapping of Course Objectives to ABET Outcomes:

Course Objectives	a) an ability to apply knowledge of mathematics, science, and engineering	(b) an ability to design and conduct experiments, as well as to analyze and interpret data	(c) an ability to design a system, component, or process to meet desired needs	(d) an ability to function on multi-disciplinary teams	(e) an ability to identify, formulate, and solve engineering problems	(f) an understanding of professional and ethical responsibility	(g) an ability to communicate effectively	(h) the broad education necessary to understand the impact of engineering solutions in a global and societal context	(i) a recognition of the need for, and an ability to engage in life-long learning	(j) a knowledge of contemporary issues	(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice
1	X										
2	X						X				
3	X	X			X					X	
4	X	X	X		X						
5	X				X						
6	X				X						
7	X		X		X						X
8	X					X				X	

Assessment Tools:

Student achievement of the course objectives is assessed by 6 bi-weekly homework assignments,5 labs, one in-class exam and a 2-hour "lab-exam".

A description of the assessments tools that were analyzed for this report is given below:
Objective 1: SSI Lab and SSI Homework – study, encode and implement decoders for one- and two-address machines.

Objective 2: In Lab 1 students are learning about RTL notation.

Objective 3: In HW2 the students must explore the various performance tradeoffs involved in CISC and RISC instruction sets and their implementations.

Objective 4: HW3, HW6 and the lab exam test the students' understanding of datapath and control unit design. The lab exam forces each student to demonstrate his or her knowledge of these areas, without help from fellow students or reference to outside resources. Each student gives a 5-minute presentation to the instructor or TA on the merits of their particular design.

Objective 5: Lab 3, Lab 4 and HW 5 investigate binary representations of integers and floating-point numbers and the instructions necessary to manipulate these representations.

Objective 6: Students develop an understanding for the memory hierarchy by examining the impact long-latency memory operations have on critical path of the datapath and control units.

Objective 7: In Lab 2 student are introduced to the assembler, linker and loader. In Labs 3 and 4, the students must design and program a floating-point multiplier and divider in MIPS assembly language. They are not allowed to use floating-point instructions.

Objective 8: In HW 4, students gain understanding of program vulnerabilities by studying the stack overflow exploit.

% of students meeting objective (**Note: "Meeting objective" was defined as obtaining over 75% of the possible points**									
Coursework	Topic	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Obj. 5	Obj. 6	Obj. 7	Obj. 8
Lab1	SSI lab (addressing modes, encoding)	89	89	89					
Lab 2	Assembler, compiler, linker, simulator							95	
Lab 3	Floating-point multiply simulator					83		83	
Lab 4	Floating-point divide simulator					100		100	
Lab 5	Datapath implementation for MIPS subset						100		
HW1	Performance			94					
HW2	SSI- operands, addressing, registers	96							
HW3	SMOK – datapath layout				100				
HW4	Procedure calls, stack overflow								100
HW5	Arithmetic, binary representation, immediate instructions					53			
HW6	Datapath and control				100				
Lab Exam	Design and implementation of small processor				96		96		96

General Evaluation in terms of Course Objectives

How well are students learning?

The students learn by doing in this class. Labs are used to give students supervised exposure to compilers, assemblers and a schematic design tool called SMOK. The number of labs seemed sufficient for most students to get the hang of using the tools. Students learned about assembly language examining output from a compiler and by writing two (2) large assembly language programs (a floating-point multiplier and a floating-point dividers). They learned about datapaths by using the SMOK tool to design an executable processor.

What issues are limiting student learning?

Large class size (over 120 students) is not ideal for classroom interaction. CS and CpE students have had some exposure to assembly language and data representation in CS216. We assume the students have had no exposure and start from the beginning. Nonetheless, the EE students do state a feeling of being disadvantaged in the course.

If changes are being made in the curriculum, how might these affect this course?

Changed the course a bit this semester. After discussions with Ron Williams about the follow-up course EE/CS435, we removed most of the pipelining discussion. We also added 1- and 2-address machines and their datapaths early in the semester.

If you have made significant changes, how have they been assessed? Have they improved learning or are further modifications warranted?

Perhaps the biggest change was the inclusion of the “lab exam”. The “lab exam” is a 2-hour exam given to each student in the class. The student must take a processor specification (ISA and simple RTL semantics) and design a processor implementation for this specification. Students were given the opportunity to make a higher grade by implementing a dual-issue version of the machine.

This “lab exam” gave the students an opportunity to use their knowledge gained over the course of the semester. It became clear which students had worked hard over the semester and learned the principles and tools. It was also clear which students had not done the work on their own as they could not finish (or in some cases, even start) the assignment in the allotted time.