

COURSE OUTLINE OF RECORD

Dept., Number	CSC 2310	Course Title	Intro. to Computer Software Systems
Semester Hours	3		
Year	2006	URL (if any):	

Current Catalog Description:

This course provides an introduction to computer architecture, systems programs, machine organization, instructions, data representation, and addressing. Topics covered include assemblers, linkers, loaders, operating systems, and elementary assembler language programming. Laboratory work required. Prerequisites: CSC 1311/CSC 3361.

Textbook:

Sivarama P. Daddamudi, Introduction to Assembly Language Programming for Pentium and RISC Processors, Springer, 2nd Edition. ISBN 0387206361

Suggested Reading:

John Waldron, Introduction to RISC Assembly Language Programming, Addison-Wesley Longman Limited, 1999. ISBN 0-201-39828-1

Web Resources:

1. The SPIM simulator (a free software simulator for running MIPS R2000 assembly language programs for Unix, DOS, and Windows). The SPIM simulator and Documentation are available at <http://www.cs.wisc.edu/~larus/spim.html>
2. Assemblers, Linkers, and the SPIM Simulator (Adobe PDF file): Appendix A of Hennessy & Patterson, Computer Organization and Design: The Hardware/Software Interface, Second Edition, Morgan Kaufmann Publishers, Inc., 1997. <http://www.cs.wisc.edu/~larus/SPIM/cod-appa.pdf>
3. MIPS Web Site. <http://www.mips.com/>

Course Goals:

1. To introduce binary, hexadecimal, and two's complement numbers and arithmetic.
2. To introduce assembly language programming.
3. To introduce computer architecture and its relation to systems programs.

Prerequisites by Topic:

Some programming experience with a higher-level language, such as C++ or Java.

Major Topics Covered in the Course (number of weeks):

- | | |
|--|-----|
| 1. Introduction (Chap 1) | 0.5 |
| 2. Data representation and computer arithmetic (Chap 2) | 2 |
| 3. MIPS computer organization (Chap 3) | 0.5 |
| 4. Introduction to MIPS assembly language programming (Chap 4) | 2 |
| 5. Control flow structures (Chap 5) | 2.5 |
| 6. Addressing modes (Chap 6) | 2 |
| 7. Logical shift and rotate instructions (Chap 7) | 0.5 |
| 8. Stacks and procedures (Chap 8) | 1 |

Laboratory Projects:

One assignment for each topic from 1 to 8

Dept., Number	CSC 2310	Course Title	Intro. to Computer Software Systems
----------------------	----------	---------------------	-------------------------------------

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Data Structures		
Software Design	1		Prog. Languages	1	
Comp. Arch.	1				

Oral and Written Communication:

Every student is required to submit at least three, (some years four) written assignments of typically one to two pages. Every student is expected to answer questions in class on these assignments, to present and explain their own solutions, and to answer questions and participate in discussions on the whole course material.

Social and Ethical Issues:

Ethical issues are studied as related to using Internet resources and, more specifically, e-mail. Guidelines are presented to students and discussed with them that can help them avoid misinterpretations, miscoordinations, mistrust, incivilities, and other problems peculiar to e-mail communication. Students are advised to establish mutual trust and respect with their e-mail partners. The class time spent on this topic is about 30 minutes. Students are required to demonstrate their understanding of this topic through an assignment or a mid-semester test.

Theoretical Content:

Binary, hexadecimal, and two's complement numbers and arithmetic - about 2.5 weeks.

Problem Analysis:

This course is both conceptual and applied. The general principles of software systems programming are taught through the study of specific assembler language, namely the Open VMS MACRO assembly language. Students are expected to analyze and understand general code patterns. Students are also expected to analyze concrete problems and to determine how the studied patterns can be applied to solve such problems.

Open-Ended Design:

Students are expected to design assembly language solutions to specific problems by using and adapting general code patterns.