COURSE DESCRIPTION

Dept., Number	**CSC 35**	Course Title	**Introduction to Computer Architecture**
Semester hours	**3**	Course Coordinator	**Kwai-ting Lan**
		URL (if any):	**http://gaia.ecs.csus.edu/~lan/**

Catalog Description

Internal representation of numeric and non-numeric data, assembly level machine architecture, addressing modes, subroutine linkage, polled input/output, interrupts, high-level language interfacing, macros and pseudo operations. Lecture two hours, technical activity and laboratory two hours. Prerequisites: CSc 15.

Textbook

Richard C. Detmer, Essentials of 80 x 86 Assembly Language, Jones and Bartlett, 2007.

Linda Null, Essentials of Computer Organization & Architecture, 2nd Edition, Jones and Bartlett, 2006.

Course Goals

1. Main objective of this course is to give students a basic understanding of the architecture/organization of a computer including machine-level I/O and interrupts.
2. To provide students with an opportunity to implement some basic data and control structures in an assembly language.

Prerequisites by Topic

Thorough understanding of:

- Basic program control structures and built-in data types.

Basic understanding of:

- Programming style and program documentation concepts.
- Program development process.
- Program design, testing, and debugging techniques.
- One-dimensional arrays.

Exposure to:

- Distinction between compiling, linking, and executing programs.
- Internal vs. external (e.g. ASCII) representation.
- Lifetime of variables.

Major Topics Covered in the Course

1. Negative number representation 1's and 2's complement, signed magnitude (1 hour).
2. Binary arithmetic (1 hour).
3. Non-numeric data representation: alphanumeric, ASCII, EBCDIC. (1/2 hour).
4. Representation of Elementary Language Data Types: integer, real, Boolean, Character (1/2 hour).
5. Basic organization: Von Neuman structure, data, address and control paths, functional units (e.g. ALU memory, registers, control unit) instructional cycle, timing, CPU cache (L1) (3 hours).
6. Instructional Types: data transfer, relative and absolute jumps, arithmetic/logic shift, subroutine linkage (including parameter passing), conditional operations/flags, and sign extension, NOP, Halt; Overview of RISC instruction sets (5 hours).
7. Assembly and machine language: overview of the assembly process (pass 1 pass 2), re-locatable code, address binding, linking and loading, macros, code vs. data space, hand assembly of instructions (4 hours).
8. Addressing modes: direct, absolute, immediate, indirect, indexed, relative (2 hours).
9. I/O: device addresses, isolated vs. memory-mapped, control and status registers, bits, polling loops, input vs. output devices, device handshaking (3 hours).
10. Interrupts: vector tables, interrupt service routines, and interrupt acknowledgment, hardware vs. software exceptions (3 hours).
11. High-level language interfacing, inline assembly, introduction to code generation (5 hours).
12. Exams (2 hours).

Outcomes

Thorough understanding of:

- Character and integer representations.
- Conversion between the different number systems, e.g., binary, octal, and hexadecimal.
- Conversion from the external string representation of the numbers to the internal representation.
- Basic instruction types and addressing modes.
- Logical truth tables for AND, OR, XOR and NOT.

Basic understanding of:

- Major functions implemented in a CPU.
- Machine instruction format.
- Instruction cycle.
- Debugging techniques at the machine level.
- Two pass assembly process.
- Serial input and output.
- Parallel handshaking and polling loops.
- Internal and external exceptions including interrupt priority.
- Hardware and software stacks.

- Memory addressability.
- Inline assembly.

Exposure to:
- Units of time measurement, CPU clocking, instruction timing.
- CPU power-on and reset operation.
- Privileged and non-privileged modes.
- Object code generation.
- Code generation.

Laboratory Projects

1. Familiarization with MASM and laboratory equipment (1 week).
2. Familiarization with Programmers workbench (1 week).
3. Program and Lab: output character data (1 week).
4. Program and Lab: subroutines and loops (1 week).
5. Program and Lab: arrays and loops (1 week).
6. Program and Lab: numeric data and arithmetic operations (1 week).
7. Program and Lab: numeric data, arrays, loops (2 weeks).
8. Program and Lab: serial I/O, polling, bit instructions (2 weeks).
9. Program and Lab: Interrupts, timer programming, software timing loops (2 weeks).
10. Program and Lab: Interrupts serial and parallel I/O (2 weeks).

Estimated Curriculum Category Content (Semester hours)

Area	*Core*	*Advanced*	*Area*	*Core*	*Advanced*
Algorithms			Data Structures		
Software Design	0.5		Prog. Languages	0.5	
Comp. Arch.	2.0				

Oral and Written Communications

No significant component.

Social and Ethical Issues

No significant component.

Theoretical Content

No significant component.

Problem Analysis

Students are required to analyze all program and laboratory problems (except the first) and determine the solution steps.

Solution Design

Students are required to do a detailed design prior to code implementation.

/sj