CS A216: COMPUTER ARCHITECTURE

Item	Value
Curriculum Committee Approval Date	12/06/2023
Top Code	070600 - Computer Science (Transfer)
Units	4 Total Units
Hours	90 Total Hours (Lecture Hours 63; Lab Hours 27)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

A course in the architecture of computers. Topics will include Boolean algebra and computer arithmetic, digital logic, micro and macro architecture, Assembly language, performance, datapath and control, memory hierarchies, interfacing and peripherals, and multiprocessing. PREREQUISITE: CS A150; and CS A100, CS A122, CS A131 or CS A170. Transfer Credit: CSU; UC.

Course Level Student Learning Outcome(s)

1. Design and analyze combinational and sequential logic circuit components of the CPU.
2. Write and analyze machine and assembly language instructions and small-scale programs.

Course Objectives

- 1. Explain what is inside the computer, the software below the programs, and the language of the computer.
- 2. Describe the internal organization of computers and how it affects program performance.
- 3. Recall the definitions of computer architecture.
- 4. Describe the different ways in which performance can be determined and the metrics for measuring performance from the viewpoint of both computer users and designers.
- 5. Recognize and write Assembly language programs suitable in speed and size to exploit HW features of a computer.
- 6. Describe the MIPS instruction set for common computer architecture, their relation to machine instruction, their HW representation, and their relationship with high-level programming languages.
- 7. Employ numbers and arithmetic algorithms and hardware features acting on them.
- 8. Demonstrate the knowledge of the basics of logic design and the logic, the combinational logic systems, truth tables, logic equations, Boolean algebra, dont cares, clocks, sequential logic systems, memory elements, registers, and RAMs.
- 9. Diagram and use finite state machines and timing methodologies.
- 10. Underline the key principles used in creating datapath and designing the control.
- 11. Recognize various implementation strategies affecting the clock rate and CPI for the machine.
- 12. Explain the concept of pipelining and various types, and graphical representations of pipelines, and pipelined controls.
- 13. Define memory hierarchies and data transfer between the adjacent levels.
- 14. Describe the types and characteristics of I/O devices, buses, I/O interface with processor and operating systems.
- 15. Discuss the key questions and models for multiprocessor design.

Lecture Content

The Basics of Logic Design Gates, truth tables, logic equations, combinational logic, clocks, finite state machines, timing methodologies, adders and subtractors, multiplexors, decoders, registers, counters Computer Abstraction and Technology Hardware parts of the computer, integrated circuits manufacturing, historical perspective The Role of Performance Defining performance, measuring performance, programs for evaluating performance Instructions: The language of the Machine Representing instructions in the computer, categories of machine instructions, number representation, Representing data structures, Assembly language and assemblers, linkers, and the SPIM simulator Arithmetic for Computers Signed and unsigned numbers, addition and subtraction, logical operations, multiplication and division, floating-point representation and arithmetic The Processor: Datapath and Control Building a datapath, implementation schemes, microprogramming, exceptions Enhancing the Performance with Pipelining Overview of pipelining, pipelined datapath and control Large and Fast: Exploiting Memory Hierarchy Basics of caches, virtual memory, memory hierarchies Interfacing Processors and Peripherals Types and characteristics of input/output devices, file systems, busses, interfacing Multiprocessors Programming multiprocessors, connecting multiprocessors to busses, network topologies

Lab Content

Students will complete a series of labs where they will learn how to write MIPS assembly language programs. Getting Started with SPIM. Students will: Load MIPS programs (assembly language) and execute them Examine memory locations Examine registers Execute programs step by step Set and remove breakpoints MIPS: The Virtual Machine. Students will learn: What synthetic instructions are family: Calibri; mso-bidi-theme-font: minor-latin; color: #242424;">How synthetic instructions expand to sequences of native instructions How MIPS addresses the memory Control Structures in MIPS. After completing the lab, students will: Know how conditional and unconditional branches work in MIPS Better understand the advantages of having fixed-sized instructions Be able to use conditional and unconditional branches in their code Register Usage and Procedures. After completing the lab, students will: Know how the MIPS integer registers are used Be able to write procedures in MIPS assembly language -theme-font: minor-latin; color: #242424;">Memory Issues. After completing the lab, students will: Have a better understanding of why memory alignment is required Know what is the difference between Big Endian and Little Endian Know how to use in MIPS assembly language programs data of sizes other than word. Arithmetic in MIPS. After completing the lab, students will: Know how to do integer arithmetic in MIPS Know how to do floating point arithmetic in MIPS Know about conversion from integer to floating point and from floating point to integer Exceptions in MIPS. After completing the lab,

students will: ; mso-bidi-theme-font: minor-latin; color: #242424;">Know the exception mechanism in MIPS Be able to write a simple exception handler for a MIPS machine

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

Lecture using PowerPoint presentations Assembly and machine language simulator (e.g. SPIM) Discussions

Reading Assignments

Students will spend a minimum of 4 hours per week reading the textbook and/or other reading material assigned.

Writing Assignments

Students will spend a minimum of 6 hours weekly completing assignments.

Out-of-class Assignments

Students will spend a minimum of 6 hours per week completing weekly assignments.

Demonstration of Critical Thinking

Written tests and quizzes Homework assignments Project and research paper evaluation Extra credit options

Required Writing, Problem Solving, Skills Demonstration

Homework assignment submissions Presentation of project work using simulator Research papers using Internet and the library about up-to-date related topics

Eligible Disciplines

Computer science: Masters degree in computer science or computer engineering OR bachelors degree in either of the above AND masters degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelors degree in engineering AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in mathematics AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in any of the above AND a masters degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Masters degree required.

Textbooks Resources

1. Required Tanenbaum, A.S.. Structured Computer Organization, 6th ed. New York: Pearson, 2020