ICS C240: COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE

Item	Value
Curriculum Committee Approval Date	11/17/2023
Top Code	070700 - Computer Software Development
Units	3 Total Units
Hours	72 Total Hours (Lecture Hours 54; Lab Hours 18)
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

This course explores the fundamental concepts of computer organization and assembly language programming. Students will dive into the inner workings of computer systems, learning about the architecture, memory hierarchy, central processing unit (CPU) design, and the role of assembly language in software development. Through hands-on exercises, students will gain practical experience in writing and debugging assembly language programs. This course serves as a bridge between low-level computer hardware and high-level software development. ADVISORY: ICS C120 and ICS C123. Transfer Credit: CSU.

Course Level Student Learning Outcome(s)

1. Identify and describe the components of the memory hierarchy, including caches, random-access memory (RAM), and secondary storage.
2. Debug and optimize assembly language programs for specific tasks.
3. Apply low-level programming techniques to solve real-world problems.

Course Objectives

- 1. Describe the architecture and organization of computer systems.
- 2. Identify and discuss the components of the memory hierarchy, including caches, RAM, and secondary storage.
- 3. Discuss and provide examples of the principles of central processing unit (CPU) design and operation.
- 4. Define the basics of assembly language programming and its role in software development.
- 5. Provide examples of how to write, debug, and optimize assembly language programs for specific tasks.
- 6. Explain how to analyze and evaluate the efficiency of assembly language code.
- 7. Show how to explore computer system and assembly language interfaces.
- 8. Describe how to apply low-level programming techniques to solve real-world problems.

Lecture Content

Introduction to Computer Organization Overview of computer architecture and organization. Role of the CPU, memory, and input/output devices. Memory Hierarchy Levels of memory (cache, RAM, secondary storage). Caching principles and memory management. Central Processing Unit (CPU) CPU components and operations. Instruction cycle and pipeline processing. Data Representation Binary, hexadecimal, and other number systems. Data encoding and representation in memory. Assembly Language Basics Introduction to assembly language. Registers, memory addressing modes, and instructions. Assembly Language Programming Writing, debugging, and testing assembly language programs. Control structures (conditional branches and loops). I/O Operations Input/output operations in assembly language. Interfacing with hardware devices. Assembly Language Optimization Techniques for code optimization and efficiency. Reducing instruction count and cycle time. Computer System Interfaces System calls and interaction with the operating system. Accessing hardware resources. Assembly Language for Real-World Applications Practical applications of assembly language in low-level system programming and embedded systems.

Lab Content

Introduction to Assembly Language: Write a simple assembly language program that displays "Hello, Assembly!" on the screen. Memory and Registers: Experiment with different registers (e.g., AX, BX, CX) and their usage. Perform memory operations such as loading and storing values. Conditional Branching: Write an assembly program that performs a conditional jump based on user input. Looping Constructs: Create a loop in assembly language to repeat a task a specified number of times. Memory Access and Addressing Modes: Experiment with different addressing modes (e.g., immediate, direct, indirect) for memory access. Subroutines and Function Calls: Write a program that uses subroutines (functions) to perform specific tasks. Input/Output Operations: Develop a program that reads user input and displays output using assembly language instructions. Assembly Language Debugging: Debug a given assembly language program with errors and identify and correct the issues.

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

This course will utilize a combination of lecture, remote virtual machine assignments, classroom/discussion student interactions, problem-solving, quizzes, tests, and troubleshooting assignments to achieve the goals and objectives of this course. All instructional methods are consistent across all modalities.

Reading Assignments

Read about the architecture and organization of computer systems. Read about the components of the memory hierarchy, including caches, RAM, and secondary storage. Read about low-level programming techniques.

Writing Assignments

Describe the components of the memory hierarchy, including caches, RAM, and secondary storage Complete hands-on coding exercises.

Out-of-class Assignments

Evaluate to debug and optimize assembly language programs for specific tasks. Discuss the principles of central processing unit (CPU) design and operation. Compare computer system and assembly language interfaces.

Demonstration of Critical Thinking

Students will be asked to read and review academic concepts in the textbook while experiencing how the CPU behaves under different conditions. Thus, practical programming techniques are taught through personal understanding, hands-on discovery, active learning, and discussing concepts and their results with others.

Required Writing, Problem Solving, Skills Demonstration

The problem-solving exercises will include analysis and evaluation of the efficiency of assembly language code. Programming skills demonstrations are included in the hands-on projects.

Eligible Disciplines

Computer information systems (computer network installation, microcomputer ...: Any bachelors degree and two years of professional experience, or any associate degree and six years of professional experience. 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 Elahi, A. Computer Systems: Digital Design, Fundamentals of Computer Architecture and Assembly Language, 1st ed. Springer, 2018 Rationale: Low-cost textbook

Other Resources

1. Technology related white papers and articles are available at no charge to all students at multiple sites as recommended by the instructor. 2. Coastline Library