GAVILAN COLLEGE
5055 Santa Teresa Blvd
Gilroy, CA 95023

Course Outline

COURSE: CSIS 28 DIVISION: 50 ALSO LISTED AS:

TERM EFFECTIVE: Spring 2021 CURRICULUM APPROVAL DATE: 12/8/2020

SHORT TITLE: COMPUTER ARCHITECTURE

LONG TITLE: Computer Architecture and Organization

Units	Number of Weeks	Type	Contact Hours/Week	Total Contact Hours
3	18	Lecture:	3	54
		Lab:	0	0
		Other:	0	0
		Total:	3	54

COURSE DESCRIPTION:

Introduction to the organization and architecture of computer systems. Mapping of statements and constructs in a high-level language onto sequences of machine instructions is studied, as well as the internal representation of simple data types and structures. Numerical computation is examined with an eye toward possible data representation errors and procedural errors. Throughout the course, students will write short assembly language programs that utilize the concepts being studied. (C-ID: COMP 142) ADVISORY: Some programming experience or programming coursework.

PREREQUISITES:

COREQUISITES:

CREDIT STATUS: D - Credit - Degree Applicable

GRADING MODES
L - Standard Letter Grade

REPEATABILITY: N - Course may not be repeated

SCHEDULE TYPES:
02 - Lecture and/or discussion
05 - Hybrid
72 - Dist. Ed Internet Delayed

STUDENT LEARNING OUTCOMES:

By the end of this course, a student should:

1. Write simple assembly language programs.

2. Discuss basic strategies and analyze design decisions of computer organization and design.

3. Demonstrate the process whereby fundamental higher level programming constructs are implemented at the machine language level.

CONTENT, STUDENT PERFORMANCE OBJECTIVES, OUT-OF-CLASS ASSIGNMENTS

Curriculum Approval Date: 12/8/2020

6 Hours

Introduction

Computer Subsystems - the Von Neumann model

How the Subsystems Interact

Data Storage Formats

Bits and Groups of Bits

Mathematical Equivalence of Binary and Decimal

Unsigned Decimal to Binary Conversion

Memory - A Place to Store Data (and Other Things)

Using C Programs to Explore Data Formats

Examining Memory With a Debugger

ASCII Character Code

Write and Read Functions

Student Performance Objectives: List the separate subsystems of computer hardware and explain how they interact. Convert an unsigned decimal integer to binary. Discuss the major components of a computer. Explore how data is encoded for storage in memory and write some programs in C to explore these concepts.

3 Hours

Computer Arithmetic

Addition and Subtraction

Arithmetic Errors - Unsigned Integers

Arithmetic Errors - Signed Integers

Overflow and Signed Decimal Integers

C/C++ Basic Data Types

Other Codes

Student Performance Objectives: Develop a good code that uses simple on/off switches to represent decimal numbers. Illustrate how truth tables work. Explore other codes such as BCD code and Gray Code and state their limitations.

3 Hours

Central Processing Unit

CPU Overview

CPU Registers

CPU Interaction with Memory and I/O

Program Execution in the CPU

Using a Debugger to View the CPU Registers

Student Performance Objectives: Discuss the programmer's view of the CPU and describe how it interacts with memory. Utilize a debugger to view the CPU registers.

6 Hours

Programming in Assembly Language

Creating a New Program

Program Organization

Viewing Both the Assembly Language Code and the Source Code

Assemblers and Linkers

Creating a Program in Assembly Language

Instructions Introduced Thus Far

Student Performance Objectives: Create a program in assembly language. Explain how assemblers and linkers work. Write a short C program in assembly language. Write assembly language functions that return an integer or return a character.

3 Hours

Program Data - Input, Store, Output

Calling write in -bit Mode

Introduction to the Call Stack

Viewing the Call Stack

Local Variables on the Call Stack

Designing the Local Variable Portion of the Call Stack

Using syscall to Perform I/O

Calling Functions, -Bit Mode

Instructions Introduced Thus Far

Student Performance Objectives: Recall how to call functions that can read input from the keyboard, allocate memory for storing data, and write output to the screen. Determine where each local variable will be located in the memory that is allocated on the call stack. Use debugger to inspect the stack and inspect argument passing registers. Modify a given so that the stack grows from lower numbered array elements to higher numbered ones. Write a program in assembly language that prompts the user to enter an integer, then displays its hexadecimal equivalent

3 Hours

Computer Operations

The Assignment Operator

Addition and Subtraction Operators

Introduction to Machine Code

Instructions Introduced Thus Far

Student Performance Objectives: Examine the most common operations - assignment, addition, and subtraction. Utilize the addition and subtraction operators.

3 Hours

Program Flow Constructs

Repetition

Binary Decisions

Instructions Introduced Thus Far

Student Performance Objectives: Demonstrate how to organize assembly language instructions to implement repetition and binary program flow constructs. Explain how to implement the binary construct. Write a program in assembly language that displays all the printable characters that are neither numerals nor letters on the screen, one character at a time. Write a program in assembly language that prompts the user to enter a text string, reads the user's input into a char array, echoes the user's input string, modifies the character in a specified way, and displays the modified string.

3 Hours

Writing Your Own Functions

Overview of Passing Arguments

More Than Six Arguments, -Bit Mode

Interface Between Functions, -Bit Mode

Instructions Introduced Thus Far

Student Performance Objectives: Describe how to use pass arguments inside the called function. Design a stack frame for a function. Complete assembly language programming assignments.

3 Hours

Bit Operations; Multiplication and Division

Logical Operators

Shifting Bits

Multiplication

Division

Negating Signed ints

Instructions Introduced Thus Far

Student Performance Objectives: Manipulate individual character codes in a text string. Write a program in assembly language that allows the user to enter a decimal integer then displays it in binary. Write a function, mul16, in assembly language that takes two 16-bit integers as arguments and returns the 32-bit product of the argument.

3 Hours

Data Structures

Arrays

Structs (Records)

Structs as Function Arguments

Structs as C++ Objects

Instructions Introduced Thus Far

Student Performance Objectives: State the main reason for organizing data into a struct format. Examine the compiler-generated assembly language for the load struct. Write a program that allows the user to maintain an address book. Write a program that allows the user to set up and maintain two bank accounts.

6 Hours

Fractional Numbers

Fractions in Binary

Fixed Point ints

Floating Point Format

IEEE

Floating Point Hardware

Comments About Numerical Accuracy

Instructions Introduced Thus Far

Student Performance Objectives: State the advantage of using a floating point format and a fixed point format. Perform both integer and floating point operations on both scalar and vector values. Explain the possible uses of the instructions introduced thus far.

6 Hours

Interrupts and Exceptions

Hardware Interrupts

Exceptions

Software Interrupts

CPU Response to an Interrupt or Exception

Return from Interrupt/Exception

The syscall and sysret Instructions

Summary

Instructions Introduced Thus Far

Student Performance Objectives: Summarize the differences between a call instruction and an interrupt/exception. List the features of the interrupt/exception. List the possible uses of the instructions. Complete assembly language programming assignments.

3 Hours

Input/Output

Memory Timing

I/O Device Timing

Bus Timing

I/O Interfacing

I/O Ports

Programming Issues

Interrupt-Driven I/O

I/O Instructions

Student Performance Objectives: Discuss the I/O subsystem. Discuss how I/O devices are programmed. Examine what each function does.

1 Hour

Review

Student Performance Objectives: Participate in review activities.

2 Hours

Final Exam

METHODS OF INSTRUCTION:

Lecture, guided discovery, video tutorials, demonstration.

OUT OF CLASS ASSIGNMENTS:

Required Outside Hours: 36

Assignment Description:

Read assigned textbook material and study for exams.

Required Outside Hours: 72

Assignment Description:

Homework: Completed assigned exercises and program assignments. Such as: Exercises related to: Binary, Decimal, Hexadecimal; Exercises related to: Two's Complement, Binary Addition; Exercises related to: Tracing a C program with a Debugger to Inspect the Registers; Assembly Language Programming Assignments: Write a short C Program in Assembly Language, Write Assembly Language Functions that Return an Integer or Return a Character; Exercises such as: Use Debugger to Inspect the Stack, Inspect Argument Passing Registers; Exercises: Read and Write Instructions. Modify a given so that the stack grows from lower numbered array elements to higher numbered ones. Write a program in assembly language that prompts the user to enter an integer, then displays its hexadecimal equivalent.; Assembly Language Programming Assignments: writeStr, ReadLn in Assembly Language, examining different levels of compiler optimization.; Assembly Language Programming Assignments: Write a program in assembly language that allows the

user to enter a decimal integer then displays it in binary. Write a function, mul16, in assembly language that takes two 16-bit integers as arguments and returns the 32-bit product of the argument.; Assembly Language Programming Assignments: Write a program that allows the user to maintain an address book. Write a program that allows the user to set up and maintain two bank accounts. Each account should have a unique account name. The user should be able to credit or debit the account.

METHODS OF EVALUATION:

Writing assignments

Percent of total grade: 20.00 %

Percent range of total grade: 5% to 20% Written Homework

Problem-solving assignments

Percent of total grade: 70.00 %

Percent range of total grade: 60% to 90% Homework Problems/Projects, Quizzes, Exams

Objective examinations

Percent of total grade: 10.00 %

Percent range of total grade: 5% to 10% Multiple Choice, True/False, Matching Items, Completion

REPRESENTATIVE TEXTBOOKS:

Robert G. Plantz. Introduction to Computer Organization with x86-64 Assembly Language&GNU/Linux. No Starch Press,2019.

Reading Level of Text, Grade: 12+ Verified by: MS Word

ARTICULATION and CERTIFICATE INFORMATION

Associate Degree:
CSU GE:
IGETC:
CSU TRANSFER:
Transferable CSU, effective 201570
UC TRANSFER:
Not Transferable

SUPPLEMENTAL DATA:

Basic Skills: N
Classification: Y
Noncredit Category: Y
Cooperative Education:
Program Status: 1 Program Applicable
Special Class Status: N
CAN:
CAN Sequence:
CSU Crosswalk Course Department: COMP
CSU Crosswalk Course Number: 142
Prior to College Level: Y
Non Credit Enhanced Funding: N
Funding Agency Code: Y
In-Service: N
Occupational Course: E
Maximum Hours:
Minimum Hours:
Course Control Number: CCC000597047
Sports/Physical Education Course: N
Taxonomy of Program: 070600