CS 12 Course Outline as of Fall 2019

CATALOG INFORMATION

Dept and Nbr: CS 12 Title: ASSEMBLY LANG PROG
Full Title: Assembly Language Programming/Computer Architecture
Last Reviewed: 1/28/2019

Units		Course Hours per Week		Nbr of Weeks	Course Hours Total	
Maximum	4.00	Lecture Scheduled	4.00	17.5	Lecture Scheduled	70.00
Minimum	4.00	Lab Scheduled	0	6	Lab Scheduled	0
		Contact DHR	0		Contact DHR	0
		Contact Total	4.00		Contact Total	70.00
		Non-contact DHR	0		Non-contact DHR	0

Total Out of Class Hours: 140.00 Total Student Learning Hours: 210.00

Title 5 Category:	AA Degree Applicable
Grading:	Grade Only
Repeatability:	00 - Two Repeats if Grade was D, F, NC, or NP
Also Listed As:	
Formerly:	CIS 22

Catalog Description:
Introductory computer architecture and techniques of assembly language programming as they apply to modern microprocessors such as I-86, ARM and/or PowerPC. Topics include theory and concepts of virtual memory, pipelines, caches, and multitasking, hardware architecture (bus, memory, stack, I/O, interrupts), design of structured assembly language code, use of software interrupts, survey arithmetic notations (binary, hexadecimal, floating- point, binary-coded decimal), input/output, and disk processing concepts.

Prerequisites/Corequisites:
Course Completion of CS 10B

Recommended Preparation:

Limits on Enrollment:

Schedule of Classes Information:
Description: Introductory computer architecture and techniques of assembly language programming as they apply to modern microprocessors such as I-86, ARM and/or PowerPC. Topics include theory and concepts of virtual memory, pipelines, caches, and multitasking,

hardware architecture (bus, memory, stack, I/O, interrupts), design of structured assembly language code, use of software interrupts, survey arithmetic notations (binary, hexadecimal, floating- point, binary-coded decimal), input/output, and disk processing concepts. (Grade Only)
Prerequisites/Corequisites: Course Completion of CS 10B
Recommended:
Limits on Enrollment:
Transfer Credit: CSU;UC.
Repeatability: Two Repeats if Grade was D, F, NC, or NP

ARTICULATION, MAJOR, and CERTIFICATION INFORMATION:

AS Degree:	**Area**			Effective:	Inactive:
CSU GE:	**Transfer Area**			Effective:	Inactive:
IGETC:	**Transfer Area**			Effective:	Inactive:
CSU Transfer:	Transferable	Effective:	Fall 1982	Inactive:	
UC Transfer:	Transferable	Effective:	Fall 1982	Inactive:	

CID:
CID Descriptor:COMP 142 Computer Architecture and Organization
SRJC Equivalent Course(s): CS12

Certificate/Major Applicable:
Major Applicable Course

COURSE CONTENT

Student Learning Outcomes:
At the conclusion of this course, the student should be able to:

1. Describe concepts of virtual memory, pipelines, caches, and multitasking, hardware architecture (bus, memory, stack, Input/Output (I/O), interrupts).
2. Apply structured assembly language code, use of software interrupts, survey arithmetic notations (binary, hexadecimal, floating- point, binary-coded decimal), input/output, and disk processing concepts.
3. Code, assemble, link, and debug Assembly Language programs, including an interrupt handler.
4. Demonstrate how fundamental high-level programming constructs are implemented at the machine-language level.

Objectives:
At the conclusion of this course, the student should be able to:

1. Distinguish and categorize the architectural components of a microcomputer.
2. Apply microcomputer design principles to identify architectural components of the Intel family of microprocessors and demonstrate ability to utilize microcomputer capabilities through assembly language programs.
3. Create a complete set of source modules using standard design tools.
4. Prepare executable assembly language programs which include at least one subroutine library module.
5. Create programs which carry out binary arithmetic operations, floating-point, and BCD (binary-coded decimal).

6. Demonstrate ability to convert numbers to and from decimal, binary, octal, and hexadecimal.
7. Use three BIOS (basic input-output system).
8. Write an interrupt handler.

Topics and Scope:

I. Assembly Language Environment
  A. Software design process
  B. Programming tools
    1. editors
    2. assemblers
    3. debuggers
    4. source modules
  C. Hardware environment
    1. networking
    2. workstations
    3. peripheral devices
  D. Assembly language overview
    1. general syntax notation
    2. instruction categories
    3. high-level language interface
    4. sub-routine library modules
II. Data Types and Number System
  A. Numeric data
    1. number system
      a. binary, decimal, octal, hexadecimal
      b. number system conversions
    2. arithmetic notation
      a. binary, signed and unsigned
      b. floating point
      c. two's complement
      d. BCD (binary-coded decimal)
  B. Character data
  C. ASCII (American Standard Code for Information Interchange) character set
III. Computer Architecture
  A. Microprocessors
  B. Data, control, address bus
  C. Registers
  D. Memory
  E. Stack
  F. Interrupts
  G. Peripheral device I/O
  H. Virtual memory
  I. Pipelines and caches
  J. CISC (complex instruction set computer) versus RISC (reduced instruction set computer)
IV. Instruction Set
  A. Addressing modes
  B. Data transfer instructions
  C. Software interrupt structure
  D. Arithmetic operations

E. Control structures
F. Stack operations
G. String operations

V. Peripheral Device Access
A. Graphics displays
B. Disk I/O
C. Standard list device

VI. Von Neumann Machine

Assignment:

1. Read approximately 25 pages per week from textbook
2. Programming exercises:
   a. Hierarchy charts and structured flowcharts
   b. Code, assemble, link, and debug approximately 10 Assembly Language programs, including an interrupt handler
3. Write technical documentation to accompany programs
4. Two to four quizzes and exams

Methods of Evaluation/Basis of Grade:

Writing: Assessment tools that demonstrate writing skills and/or require students to select, organize and explain ideas in writing.

Written documentation	Writing 0 - 10%

Problem Solving: Assessment tools, other than exams, that demonstrate competence in computational or non-computational problem solving skills.

Programming exercises	Problem solving 40 - 60%

Skill Demonstrations: All skill-based and physical demonstrations used for assessment purposes including skill performance exams.

None	Skill Demonstrations 0 - 0%

Exams: All forms of formal testing, other than skill performance exams.

Quizzes and exams	Exams 40 - 60%

Other: Includes any assessment tools that do not logically fit into the above categories.

Attendance and participation	Other Category 0 - 10%

Representative Textbooks and Materials:

x86-64 Assembly Language Programming with Ubuntu (1.1.14). Jorgensen, Ed. 2018

Introduction to Computer Organization: ARM Assembly Language Using the Raspberry Pi. Plantz, Robert. 2017