Las Positas College
3000 Campus Hill Drive
Livermore, CA 94551-7650
(925) 424-1000
(925) 443-0742 (Fax)

Course Outline for CS 21

COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING

Effective: Fall 2017

I. CATALOG DESCRIPTION:
CS 21 — COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE PROGRAMMING — 4.00 units

Basics of machine architecture, cpu architecture and design, machine language, assembly language, operating system and higher level language interface. Data representation, instruction representation and execution, addressing techniques and use of macros. Space and time efficiency issues. Input/output including video modes. Procedures including parameter passing and linkage to higher level languages.

3.00 Units Lecture 1.00 Units Lab

Prerequisite
CS 1 - Computing Fundamentals I
with a minimum grade of C

Grading Methods:
Letter or P/NP

Discipline:

	MIN
Lecture Hours:	54.00
Lab Hours:	54.00
Total Hours:	108.00

II. NUMBER OF TIMES COURSE MAY BE TAKEN FOR CREDIT: 1

III. PREREQUISITE AND/OR ADVISORY SKILLS:

Before entering the course a student should be able to:

A. CS1
1. Design, create and compile C++ programs within multiple development environments and operating systems, including the use of command-line tools in Unix/Linux.
2. Interpret and apply C++ control structures for sequencing, selection and iteration.
3. Interpret and implement programmer-defined functions in C++.
4. Create and interpret expressions involving arithmetic and logical operators;
5. Interpret and apply arrays and simple programmer-defined data structures in C++.
6. Modify and expand short programs that use standard conditional and iterative control structures and functions.
7. Choose appropriate conditional and iteration constructs for a given programming task.
8. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
9. Analyze and explain the behavior of simple programs.
10. Describe, interpret and apply the mechanics of parameter passing.
11. Discuss and apply the concept of algorithms in problem-solving processes.
12. Judge the correctness and quality of algorithms, identifying necessary properties of good algorithms.
13. Describe and apply effective debugging strategies.
14. Identify properties of variables and apply different forms of variable binding, visibility, scoping, and lifetime management.
15. Explain, interpret and apply elements of syntax related variable types, including type-checking, abstraction, type incompatibility and type safety.
16. Design, implement, test, and debug programs using basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

IV. MEASURABLE OBJECTIVES:

Upon completion of this course, the student should be able to:

A. Explain computer and CPU architecture/design and operating system interface
B. Describe and develop/explain example scenarios of the implementation of cpu based instruction pipelining and how it can be used to increase instruction throughput by performing multiple operations at the same time
C. Describe and discuss the use of CPU cache and how they are used to reduce both time and energy costs within the CPU and memory
D. Use virtual machine technology (VM) to develop both windows and linux based assembly solutions and explain how the CPU handles this virtual machine technology
E. Demonstrate machine addressing techniques

F. Demonstrate how data is represented in the machine: integers, characters, strings, floating point numbers (in IEEE representation), arrays
G. Handle integer arithmetic and string manipulations efficiently
H. Program selection and repetition constructs, macros and procedures in assembly language including parameter passing and linkage to both external assembly language modules and higher level language modules
I. Program basic keyboard input and text screen output using operating system interrupts and library procedures
J. Perform output in graphics screen modes
K. Explain how floating point arithmetic is performed using the co-processor
L. Program elementary file input/output in assembly language
M. Program interupt handlers which interface with both internal and external computing devices

V. CONTENT:

A. Computer architecture
   1. The architecture of the central processor
   2. Central processor registers
   3. Communication with memory
   4. Memory cache and its impact on performance
   5. Instruction format and execution cycle
B. Addressing techniques
   1. Absolute vs relative addressing
   2. Indexed addressing
   3. Indirect addressing
C. Machine and assembler representation of data
   1. Base arithmetic and base conversion
   2. Integers of byte and word size and larger
   3. Characters, character strings and arrays
   4. Floating point numbers in IEEE format
D. Assembly language programming fundamentals
   1. Assembly format including labels, operation codes, operands and remarks
   2. The assembly process and production/meaning of source, object, listing, map and executable files
   3. Pseudo ops
   4. Machine format of instructions
   5. Use of the registers and moving data to/from memory
   6. Integer arithmetic and logical machine instructions
   7. Basic input from keyboard and output to screen of integers, characters and strings using DOS interrupts and library routines and prewritten macros
   8. Test and compare instructions and use in selection constructs
   9. Branch, jump and loop instructions and use in repetition constructs
   10. Index register modified instructions and use with arrays
   11. Procedures which do not explicitly pass parameters
E. Advanced assembly language topics
   1. Procedures including parameter passing using the stack
   2. and stack/base pointer registers
   3. Writing macros
   4. Interupts and Interupt handlers
   5. Input and output to files
   6. Video modes
   7. Floating arithmetic using the co-processor
   8. Advanced use of segments and segment registers
   9. Instrution pipelining and increased CPU throughput
   10. How an assembler is written
F. Program development
   1. Error detection using on line debugging programs
   2. Automating the assembly, link, debug cycle
   3. Program linkage with externally assembled procedures
   4. Program linkage with higher level languages
   5. Virtual Machines(VM) and their impact on the CPU

VI. METHODS OF INSTRUCTION:

A. **Lecture** -
B. **Demonstration** -
C. **Projects** - Optional: Programming projects completed in teams
D. **Lab** - Lab Programming Assignments
E. **Discussion** -

VII. TYPICAL ASSIGNMENTS:

A. Write an assembly program to input two integers. Compute and display the sum, difference, and average of the two numbers.
B. Write an assembly program to input a string of characters and display them backwards.
C. Write an assembly program to input a set of numbers from a data file and display the average of the numbers. Be sure to handle the case where there are no numbers in the file. This should be done for a file of integers and for a file of real numbers.

VIII. EVALUATION:

A. **Methods**

   1. Other:
      a. Programming Assignments
      b. Written Homework
      c. Exams and/or Quizzes
      d. Final Examination

B. **Frequency**

   1. Programming assignments to cover each topic within the course content (contents can be combined). At least two of the programming assignments must be in-class, or covered by in-class examinations. It is suggested that there be at least one programming assignment each week.
   2. At least two in-class midterm examinations, or one in-class midterm examination and several quizzes
   3. Additional midterm examination(s) and/or quizzes
   4. In-class comprehensive final examination

IX. TYPICAL TEXTS:

1. Duntemann, Jeff. *Assembly Language Step-by-Step: Programming with Linux*. 3rd ed., Wiley, 2009.
2. Irvine, Kip. *Assembly Language for Intel-based Computers:.* 7th ed., Prentice Hall, 2014.
3. Mazidi, Muhammad . *ARM Assembly Language Programming & Architecture.* 2nd ed., Mazidi & Naimi, 2016.

X. OTHER MATERIALS REQUIRED OF STUDENTS:

A. USB Memory Device