College of San Mateo
Official Course Outline

1. **COURSE ID:** CIS 264 **TITLE:** Computer Organization and Systems Programming **C-ID:** COMP 142
   Units: 4.0 units **Hours/Semester:** 48.0-54.0 Lecture hours; 48.0-54.0 Lab hours; and 96.0-108.0 Homework hours
   Method of Grading: Grade Option (Letter Grade or Pass/No Pass)
   Prerequisite: MATH 120, and CIS 254, or CIS 278

2. **COURSE DESIGNATION:**
   Degree Credit
   Transfer credit: CSU; UC
   AA/AS Degree Requirements:
   CSM - COMPETENCY REQUIREMENTS: C1 Math/Quantitative Reasoning Basic Competency

3. **COURSE DESCRIPTIONS:**
   Catalog Description:
   The internal organization and operation of digital computers. Systems programming in C. Assembly languages, machine architecture, support for high-level languages (logic, arithmetic, instruction sequencing) and operating systems (I/O, interrupts, memory management, process switching). Elements of computer logic design. Tradeoffs involved in fundamental architectural design decisions.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**
   Upon successful completion of this course, a student will meet the following outcomes:
   1. Write and debug assembly programs that use load/store, arithmetic, logic, branches, call/return and push/pop instructions.
   2. Demonstrate how fundamental high-level programming constructs are implemented at the machine-language level.
   3. Write C programs involving Pointers, Arrays, Strings and demonstrate C Memory Management.
   4. Demonstrate how caching works in computer systems.
   5. Explain Compilation, Linking and Loading processes, Thread-Level Parallelism, Pipelining and Virtual Memory.

5. **SPECIFIC INSTRUCTIONAL OBJECTIVES:**
   Upon successful completion of this course, a student will be able to:
   1. Perform arithmetic on twos-complement integer data, expressing result in binary, hex and signed decimal.
   2. Code execute and debug assembly language programs using a target instruction set.
   3. Program selection and repetition constructs, procedures and macros in assembly language.
   4. Demonstrate parameter passing mechanisms and linkage to both external assembly language modules and higher-level language modules.
   5. Enumerate the steps used to translate a compiled higher-level language to a functioning executable program.
   6. Code C programs involving Pointers, Arrays and Strings.
   7. Understand C Memory Management.
   8. Understand Cache Memory, Virtual Memory and address translation.
   9. Understand Compilation, Linking, and Loading a program in Computer Systems.
   10. Understand Thread-Level Parallelism.
   11. Describe Synchronous Digital Systems and Combinational Logic Design.

6. **COURSE CONTENT:**
   Lecture Content:
   1. Number Representation and Floating point
   2. Basic computer organization of von Neumann machines
   3. Control unit: instruction fetch, decode, and execution
   4. Instruction Sets and types (data manipulation, control, and I/O)
   5. C programing, pointer, arrays and C memory management
   6. Assembler/Linker
   7. Assembly/machine languages programming
   8. SIMD Instructions.
   9. Assembly Function/procedure calls and return mechanisms
   10. I/O and interrupts
   11. Cache Memory

12. Virtual Memory
13. Compilation, Assembly, Linking and Loading
14. Thread-Level parallelism
15. Introduction to Synchronous Digital Systems
16. Combinational Logic Design
17. Single-Cycle CPU Datapath
18. Pipelining

Lab Content:

1-Working with instructions involving different size registers (AH, AX, AL, EAX for example) and using the debugger to step through the instructions
2- Defining and initializing various variable types (byte, word, dword, text string, and array)
3- Converting C program to Assembly
4- Branching in Assembly
5- Using shift, rotate, and the bitwise and, or, not, xor
6- Accessing data by address
7- Working with procedure both when input arguments are in registers or on the stack
8- Working with two dimensional array in Assembly
9- Pointers and Arrays in C
10- Memory allocation in C
11- Void Pointer and Function Pointer in C
12- Embedding Assembly in C
13- Optimizing program performance
14- Running program on a system; Linking. Exception Control Flow and Virtual Memory
15- Interaction and communication between programs
16- Socket programming

7. **REPRESENTATIVE METHODS OF INSTRUCTION:**

Typical methods of instruction may include:

A. Lecture
B. Lab
C. Activity
D. Discussion
E. Guest Speakers
F. Other (Specify): In-class group projects.

8. **REPRESENTATIVE ASSIGNMENTS**

Representative assignments in this course may include, but are not limited to the following:

Writing Assignments:

All programming projects (below) will be fully documented to allow clarity to the reader.

Reading Assignments:

Students will read all chapters of the required textbook, readings parallel current project and lecture content.

Other Outside Assignments:

There will be 8 programming assignments.

PROJECT 1 Number conversion worksheet, covering binary, octal, hex representations, and 2's complement arithmetic. High level program which converts decimal input to another base of choice.

PROJECT 2 Sequential instruction programming project which utilizes arithmetic instructions.

PROJECT 3 Programming project which utilizes jump instructions to build conditional structures.

PROJECT 4 Programming project which utilizes looping structures, direct and indirect addressing.

PROJECT 5 Programming project requiring at least one procedure definition/call, one function definition/call and a macro.

PROJECT 6 Programming project using array storage for a numeric type, involving search, update and output of collection.

PROJECT 7 Programming project involving string input, parsing and character type conversion.

PROJECT 8 Final non-trivial team project, satisfying all objectives of previous projects.

9. **REPRESENTATIVE METHODS OF EVALUATION**

Representative methods of evaluation may include:

A. Class Participation
B. Class Performance
C. Class Work
D. Exams/Tests
E. Group Projects
F. Homework
G. Lab Activities
H. Projects
I. Quizzes

10. **REPRESENTATIVE TEXT(S):**

Possible textbooks include:

A. Bryant, Randal E.. *Computer Systems, A programmer's Perspective Paperback*, Third ed. Pearson, 2018
B. David Patterson. *Computer Organization and Design RISC-V edition*, 2 ed. Morgan Kaufmann, 2020
C. Harris, Sarah and Harris, David. *Digital Design and Computer Architecture Risc-V edition*, 1 ed. Elsevier/Morgan Kaufmann,, 2021
D. Kernighan, Brian and Ritchie, Dennis. *The C Programming Language*, 2 ed. Prentice Hall, 1988
E. Null, Linda, and Julia Lobur. *The Essentials of Computer Organization and Architecture*, 5 ed. Burlington, MA: Jones & Bartlett, 2018

Origination Date: October 2021
Curriculum Committee Approval Date: November 2021
Effective Term: Fall 2022
Course Originator: Kamran Eftekhari