College of San Mateo
Official Course Outline

1. **COURSE ID:** CIS 264 **TITLE:** Computer Architecture and Assembly Language **C-ID:** COMP 142
   Units: 3.0 units **Hours/Semester:** 48.0-54.0 Lecture 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
   Recommended Preparation:
   Eligibility for ENGL 838 or ENGL 848 or ESL 400.

2. **COURSE DESIGNATION:**
   Degree Credit
   Transfer credit: CSU; UC

3. **COURSE DESCRIPTIONS:**
   Catalog Description:
   This course covers the internal organization and operation of digital computers at the assembly language level. Topics include the mapping of high-level language constructs into sequences of machine-level instructions; assembly language and assemblers; linkers and loaders; internal data representations and manipulations; numerical computation; input/output (I/O) and interrupts; functions calls and argument passing; and the basic elements of computer logic design. A materials fee as shown in the Schedule of Classes is payable upon registration.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**
   Upon successful completion of this course, a student will meet the following outcomes:
   1. Describe the architectural components of a computer system.
   2. Convert between decimal, binary, and hexadecimal notations.
   3. Write simple assembly language program segments.
   4. Write and debug assembly programs that use load/store, arithmetic, logic, branches, call/return and push/pop instructions.
   5. Demonstrate how fundamental high-level programming constructs are implemented at the machine-language level.

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. **COURSE CONTENT:**
   Lecture Content:
   1. Bits, bytes, and words
   2. Numeric data representation and systems
   3. Fixed and floating point systems
   4. Signed and two-complement systems
   5. Representation of nonnumeric information
   6. Representation of records and arrays
   7. Basic computer organization of von Neumann machines
   8. Control unit: instruction fetch, decode, and execution
   9. Instruction sets and types (data manipulation, control, and I/O)
   10. Assembler/Linker
   11. Assembly/machine language programming
   12. Instruction formats
   13. Addressing modes
   14. Function/procedure calls and return mechanisms

15. I/O and interrupts
16. Logic gates and storage elements
17. Logic circuit design
18. Logic design components
19. Addition/subtraction logic
20. Multiplication logic
21. Division logic

7. **REPRESENTATIVE METHODS OF INSTRUCTION:**
Typical methods of instruction may include:
   A. Lecture
   B. Activity
   C. Discussion
   D. Observation and Demonstration
   E. Other (Specify): Lectures, to introduce new topics: “Models” for problem-solving techniques; Class (group) problem solving , each person contributing a potential "next step"; Student participation in short in-class projects; Q/A sessions with students providing both the questions AND the answers; Students working in small groups to solve significant programming assignments. Live code development/debugging demonstrations

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. Homework
   F. Projects
   G. Quizzes

10. **REPRESENTATIVE TEXT(S):**
Possible textbooks include:

A. Null, Linda, and Julia Lobur. *The Essentials of Computer Organization and Architecture*, Fourth Edition ed. Burlington, MA: Jones & Bartlett, 2015
B. Plantz, Robert G. *Introduction to Computer Organization with x86-64 Assembly Language & GNU/Linux*, ed. Santa Rosa: Robert G Plantz, 2015
C. Bryant, Randal E.. *Computer Systems, A programmer's Perspective*, Third ed. Pearson, 2016
D. David Patterson. *Computer Architecture*, 6 ed. Morgan Kaufmann, 2017

Origination Date: April 2017
Curriculum Committee Approval Date: September 2018
Effective Term: Fall 2019
Course Originator: Kamran Eftekhari