GROSSMONT COLLEGE

 COURSE OUTLINE OF RECORD

Curriculum Committee Approval: 11/30/2021

 GCCCD Governing Board Approval: 12/14/2021

COMPUTER SCIENCE INFORMATION SYSTEMS 165 – ASSEMBLY LANGUAGE AND MACHINE ARCHITECTURE

 1. Course Number Course Title Semester Units

 CSIS 165 Assembly Language and Machine Architecture 4

Semester Hours

3 hours lecture: 48-54 hours 96-108 outside-of-class hours for lecture 3 hours lab: 48-54 hours

192-216 total hours

 2. Course Prerequisites

 A “C” grade or higher or “Pass” in CSIS 293 OR CSIS 296 or equivalent

 Corequisite

 None

 Recommended Preparation

 None

 3. Catalog Description

This is an introductory course in assembly language programming and machine architecture for small computers. Topics covered include number theory, registers, memory, CPU, linkers, debuggers, basic language syntax and high-level language/operating system interface. This course is intended for persons with a prior background in any other programming language and will emphasize those applications not easily performed using higher-level languages.

 4. Course Objectives

 The student will:

 a. Analyze from a software and user’s point of view the various architectural components of the specific small computer systems used in this class.

 b. Design programs for the computer using major machine and assembly language commands.

 c. Create simple modifications or extensions to existing operating system structures to provide additional system capabilities.

 d. Choose specialized commands as needed to provide further capabilities.

 e. Practice and demonstrate ability to apply knowledge of assembly language programming and machine architecture.

 f. Examine the relationships between fundamental high level programming and assembly and machine level language conversion.

 5. Instructional Facilities

 A classroom with at least one microcomputer workstation per student.

 6. Special Materials Required of Student

 Electronic storage media.

7. Course Content

 a. System architecture and overall operation.

 b. Numbering systems: decimal, binary, and hexadecimal.

 c. I/O units, operation and data representation.

 d. Coding systems used.

 e. CPU architecture.

 f. I/O bus structure.

 g. Registers, instructions and data flow.

 h. Instruction formats, machine and assembler.

 i. Internal data storage representation.

 1) Little endian.

 2) Big endian.

 3) Signed data representation.

 4) IEEE floating point.

 j. Basic instructions.

 1) Main memory.

 2) Arithmetic.

 3) Logic.

 4) I/O.

 k. Loaders and linkages.

 l. Running the assembler.

 m. Segments: stack and stack frame, code, data.

 n. High level language interface (such as C).

 o. Public ad external data in both assembler and high level language.

 p. Public procedures and functions in both assembler and high level language.

 q. Parameter passing between assembler and high level language in both directions.

 r. Macro commands and macro development both assembler and high level language.

 s. Advance I/O.

 t. Timing considerations.

 u. More advanced instructions for specialized functions.

 v. Relationships between fundamental high level programming and assembly and machine level language conversion.

 8. Method of Instruction

 a. Lecture with demonstrations.

 b. Hands-on practice.

 9. Methods of Evaluating Student Performance

 a. In-class written quizzes.

 b. Projects with program documentation.

 c. Final objective examination written or performance.

10. Outside Class Assignments

 Assembly language programming projects. Examples would include writing a program to solve a business problem, a scientific problem, or an interactive game situation.

11. Representative Texts

 a. Representative Texts:

Irvine, Kip. *Assembly Language for x86 Processors*. 8th edition. New York, NY, Pearson Publishing, 2020

 b. Supplementary texts and workbooks:

 None.

 Addendum: Student Learning Outcomes

Upon completion of this course, our students will be able to do the following:

a. Design, write, and test programs for the computer using assembly language

b. Extend C++ programs using assembly language procedures and data.