Course Title: Computer Organization and Assembly Language
Catalog Number: CSC175
Semester Hour Credit: 4.0
Prerequisite: CSC142

I. COURSE DESCRIPTION
Introduces hardware organization, memory addressing, data conversions, data representation, assembly language versus machine language, use of condition tests, branches, loops and arrays, subroutines and parameter passing.

II. GENERAL COURSE OBJECTIVES
Upon completion of this course, students will be able to:
- demonstrate proficiency in programming using an assembly language.
- demonstrate proficiency in the use of an assembler and related tools.
- explain the language translation process.
- understand the relationship between high level programming languages and machine level implementation.
- appreciate the concept of a computer system as a construct built upon many layers of abstraction.
- understand computer architecture and its relationship to higher level machine abstractions.

III. COURSE CONTENT AND SPECIFIC OBJECTIVES
A. Higher Level Language Model
   - Describe the nature of computation from the higher level language perspective.
   - Identify and describe attributes common to higher level languages, including:
     - data and data types
     - executable statements
     - control structures & program flow
     - modularity
     - higher level language as protector

B. Assembly Language Model
   - Identify and describe the functional components of a computer system (processor, storage and input/output) in terms of assembly language commands.
   - Manipulate data in assembly language code.
   - Create structured programs in assembly language.
   - Assemble, debug and execute assembly language programs.
   - Explain how computers represent integers, real numbers, and character data.
     - representation of negative numbers
     - storage capacity and its effect on numeric magnitude
   - Perform arithmetic operations on binary and hexadecimal notations
   - Convert numbers between decimal, binary and hexadecimal notations.
   - Use condition codes to implement control structures in assembly language.
condition code register
simple & compound conditionals

- Represent data structures in assembly language
  pointers and arrays
  stacks and queues
  records
  linked lists

C. Machine Language Model
- Describe the fetch/execute style.
- Explain the parts of a machine language instruction.
  op code
  operand(s)
  mode
- Identify and explain the functions of special-purpose registers.
  program counter
  instruction register
  stack pointer
- Explain the role of the bus.
- Describe the relationship between machine language instructions and assembly language instructions, directives and macros.
- Explain the nature of run-time errors.
- Explain the advantages of two-pass assemblers.
- Explain the processes of linking and loading.
- Link high-level language procedures to assembly language programs and vice-versa.
- Describe relative and absolute addressing.
- Describe the process of conditional assembly.

D. The Digital Logic Model
- Interpret and create truth tables describing the operations of Boolean algebra.
- Combine simple Boolean operations to represent logical expressions.
- Describe how Boolean logic is used for program control in assembly language.
- Perform bit-mask operations in assembly language.
- Use bit-shuffling to perform arithmetic operations.
- Diagram digital circuits using and-gates, or-gates, xor-gates and not-gates.
- Describe the construction of a computer processor in terms of digital circuits.
  bus
clock
flip-flops
decoders
multiplexers
half-adders & adders
shifters
- Describe how transistors and integrated circuits are constructed using nand- and nor-gates.
- Describe memory read and write operations.
- Describe the interactions of primary memory with the fetch/execute cycle, the clock and the bus.
- Identify and describe the characteristics of various types of memory.
• Describe the properties, uses and logical organization of secondary memory.
• Distinguish between random and sequential access methods and describe their implications.
• Explain the use of parity for error detection.
• Describe the use of ports, buffers and interrupts for input/output.

IV. INSTRUCTIONAL FORMAT
Class meets four hours per week. Approximately two-thirds of class time will be devoted to lecture/discussion and one-third will be spent in a structured lab environment performing exercises to enhance students’ understanding and apply the lecture material. Both individual and group lab exercises will be used.

V. STUDENT EVALUATION
Course grade will be based on total points accumulated from a combination of three one-hour exams, programming projects, and in-class exercises. Exam points will constitute about 60% of the grade and projects and exercises will account for the remaining 40%. Exams will be designed to test the students’ grasp of the lecture material and will include short answer, problem solving, code interpretation and code writing.

VI. RESOURCES

Supplemental Resources:
• Intel-based personal computers.
• PC Assembler, Linker, Debugger and Editor.
• Instructor-prepared handouts and course web page.