Modesto Junior College
Course Outline of Record
CMPSC 241

I. OVERVIEW
The following information will appear in the 2010 - 2011 catalog

CMPSC 241 Assembly Language Programming 4 Units
Prerequisite: Satisfactory completion of CMPSC 205.

First course in computer architecture and assembly language programming. Data representation and manipulation, CPU organization and memory, addressing modes, logic and control, table processing, and I/O control processes will be examined. Macros, program modules, and interrupts will be studied. Extensive hands-on computer projects implementing course objectives will be assigned.

Field trips might be required. (A-F or P/NP - Student choice) Lecture /Lab
Transfer: (CSU, UC) General Education: (MJC-GE: D2)

II. LEARNING CONTEXT
Given the following learning context, the student who satisfactorily completes this course should be able to achieve the goals specified in Section III, Desired Learning:

A. COURSE CONTENT

1. Required Content:
   a. Introduction to PC Hardware
      i. Bits and Bytes
      ii. Binary, Decimal, Octal, and Hexadecimal Number Systems
      iii. Character Code Representations
      iv. The Central Processing Unit (CPU)
      v. Internal Memory and Memory Maps
      vi. Segments and Addressing
      vii. Hardware Registers
   
   b. PC Software Requirements
      i. Operating System Characteristics
      ii. The Boot Process and The System Program Loader
      iii. DOS-BIOS Interface
      iv. The Stack, Program Addressing, Memory and Register References

   c. Assembly Language Requirements
      i. Editors, Assemblers, and Linkers
Assemblers versus Interpreters or Compilers

Assembly Language Syntax

Statements and Definitions

Data and Segment Directives

d. Assembling, Linking, and Executing

Assembly Language Source Code and .LST Listings

Simple Segment Directives versus Conventional Segment Directives

Two-Pass Assembler

Error Diagnostics and Debuggers

Linking Object Code and Program Execution

Cross-Reference Listings

e. Processor Instructions and Addressing

The CPU Instruction Set

Operators and Operands

Alignment of Addresses

Program Segments and Near and Far Addresses

The Segment Override Prefix

f. Program Logic and Control

The Flags Register

Instruction Labels and Short, Near and Far Addresses

Unconditional and Conditonal Jumps, Jump Tables, and Loop Control

Data Comparison, Boolean Operations, Shifting and Rotating Bits

Procedure Calls

g. Screen and Keyboard Processing

Cursor Control and Setting Screen Attributes

DOS and BIOS Interrupts, Controlling Screen and Keyboard I/O and Display Modes

Video Adapters, Buffers, and Pages

Standard File Handles for Screen Display and Keyboard Input

h. Data Manipulation

String Operations
ii. Processing Binary Data, Unsigned and Signed Arithmetic Processes

iii. Processing ASCII and BCD Data

iv. Table Processing

i. External I/O Processing
   i. Disk Storage Organization
   ii. Creating and Accessing External Files
   iii. DOS Interrupt Operations for Supporting Disks and Files
   iv. BIOS Interrupt Disk Operations
   v. Printing and Other I/O Processes

j. Advanced Programming
   i. Writing MACROS
   ii. Subprograms and Separately Assembled Program Segments
   iii. EXTRN and PUBLIC Attributes
   iv. Common Data in Subprograms
   v. Passing Parameters
   vi. Intrasegment and Intersegment Calls
   vii. Linking External Libraries
   viii. Program Interrupts

2. **Required Lab Content:**

   a. Introduction to PC Hardware
      i. Bits and Bytes
      ii. Binary, Decimal, Octal, and Hexadecimal Number Systems
      iii. Character Code Representations
      iv. The Central Processing Unit (CPU)
      v. Internal Memory and Memory Maps
      vi. Segments and Addressing
      vii. Hardware Registers

   b. PC Software Requirements
      i. Operating System Characteristics
      ii. The Boot Process and The System Program Loader
iii. DOS-BIOS Interface
iv. The Stack, Program Addressing, Memory and Register References

c. Assembly Language Requirements
i. Editors, Assemblers, and Linkers
ii. Assemblers versus Interpreters or Compilers
iii. Assembly Language Syntax
iv. Statements and Definitions
v. Data and Segment Directives

d. Assembling, Linking, and Executing
i. Assembly Language Source Code and .LST Listings
ii. Simple Segment Directives versus Conventional Segment Directives
iii. Two-Pass Assembler
iv. Error Diagnostics and Debuggers
v. Linking Object Code and Program Execution
vi. Cross-Reference Listings

e. Processor Instructions and Addressing
i. The CPU Instruction Set
ii. Operators and Operands
iii. Alignment of Addresses
iv. Program Segments and Near and Far Addresses
v. The Segment Override Prefix

f. Program Logic and Control
i. The Flags Register
ii. Instruction Labels and Short, Near and Far Addresses
iii. Unconditional and Conditional Jumps, Jump Tables, and Loop Control
iv. Data Comparison, Boolean Operations, Shifting and Rotating Bits
v. Procedure Calls

g. Screen and Keyboard Processing
i. Cursor Control and Setting Screen Attributes
ii. DOS and BIOS Interrupts, Controlling Screen and Keyboard I/O and Display Modes
iii. Video Adapters, Buffers, and Pages
iv. Standard File Handles for Screen Display and Keyboard Input

h. Data Manipulation
   i. String Operations
   ii. Processing Binary Data, Unsigned and Signed Arithmetic Processes
   iii. Processing ASCII and BCD Data
   iv. Table Processing

i. External I/O Processing
   i. Disk Storage Organization
   ii. Creating and Accessing External Files
   iii. DOS Interrupt Operations for Supporting Disks and Files
   iv. BIOS Interrupt Disk Operations
   v. Printing and Other I/O Processes

j. Advanced Programming
   i. Writing MACROS
   ii. Subprograms and Separately Assembled Program Segments
   iii. EXTRN and PUBLIC Attributes
   iv. Common Data in Subprograms
   v. Passing Parameters
   vi. Intrasegment and Intersegment Calls
   vii. Linking External Libraries
   viii. Program Interrupts

B. ENROLLMENT RESTRICTIONS

1. Prerequisites
   Satisfactory completion of CMPSC 205.

2. Requisite Skills
   Before entering the course, the student will be able to:
   a. Computer Science background equivalent to ACM CS 1 course.

C. HOURS AND UNITS
### D. METHODS OF INSTRUCTION (TYPICAL)

*Instructors of the course might conduct the course using the following method:*

1. Lecture and Discussion Methods
2. Classroom Demonstrations
3. Technology Presentations
4. Question and Answer Sessions
5. Lab Demonstrations
6. Practical Hands-On Exercises
7. Independent Study through readings

### E. ASSIGNMENTS (TYPICAL)

1. **EVIDENCE OF APPROPRIATE WORKLOAD FOR COURSE UNITS**

   *Time spent on coursework in addition to hours of instruction (lecture hours)*

   a. (Weekly) Categorize and analyze assigned topic readings.
   b. (Weekly) Carefully analyze and follow detailed instructions for completion of assignments.
   c. (Weekly) Quizzing on problem solving and programming concepts and terminology.
   d. (Weekly) Lab activities utilizing computing technology to apply software engineering and programming concepts.
   e. (Weekly) Design algorithmic solutions based on programming design and software engineering concepts.
   f. (Per term) Several exams at strategic points during the term.

2. **EVIDENCE OF CRITICAL THINKING**

   *Assignments require the appropriate level of critical thinking*

   a. Assignment Question: Under what circumstances does the DAA instruction set the Carry flag? Give an example.
   b. Assignment Question: Explain why overflow cannot occur when the MUL and one-operand IMUL instructions execute.
   c. Lab Project: The greatest common divisor of two integers is the largest integer that will evenly divide both integers. The GCD algorithm involves integer division in a loop. Design and implement this function in assembly language and write a test program that calls the function several times, passing it different values. Display all results on the screen.
   d. Lab Project: Write a procedure named PackedToAsc that converts a 4-byte packed decimal integer to a string of ASCII decimal digits. Pass the packed integer and the address of a buffer holding the ASCII digits to the procedure. Write a short test program that displays several converted integers.
e. **Example Quiz/Exam Questions**
   
i. Answer true or false: It is possible to define a procedure inside an existing procedure.
   
ii. Answer true or false: The LOOPE instruction jumps to a label when (and only when) the Zero flag is clear.
   
iii. Answer true or false: INT 21h Function 7305h reads one or more disk sectors only in protected mode.
   
iv. What is the primary advantage to using macros with parameters versus macros without them?
   
v. Write a macro named mGenRandom that generates a random integer between 0 and n-1. Let n be the only parameter.

---

**F. TEXTS AND OTHER READINGS (TYPICAL)**


---

**III. DESIRED LEARNING**

**A. COURSE GOAL**

*As a result of satisfactory completion of this course, the student should be prepared to:*

articulate the hardware architecture and component functions for a modern microprocessor and systems board; articulate and utilize a processor instruction set to construct assembly language programs; and construct basic and advanced manipulation and I/O processes within assembly language programs.

**B. STUDENT LEARNING GOALS**

*Mastery of the following learning goals will enable the student to achieve the overall course goal.*

1. **Required Learning Goals**

   *Upon satisfactory completion of this course, the student will be able to:*

   a. Demonstrate conversions between and arithmetic operations on various number system representations.
   
   b. Describe the hardware architecture and component functions for a modern microprocessor and systems board.
   
   c. Describe and demonstrate usage of various memory addressing modes.
   
   d. Explain the purpose of editors, assemblers, linkers, loaders, and operating systems runtime requirements.
   
   e. Compare and contrast assemblers, compilers, and interpreters.
   
   f. Discuss memory and program segmentation.
   
   g. Describe and utilize a processor instruction set to construct assembly language programs.
   
   h. Demonstrate program logic and control structures within assembly language programs.
   
   i. Demonstrate basic and advanced data manipulation and I/O processes within assembly language programs.
   
   j. Implement advanced assembly language processes through MACROS, program modules, and
2. **Lab Learning Goals**  
*Upon satisfactory completion of the lab portion of this course, the student will be able to:*

a. Perform conversions between and arithmetic operations on various number system representations.

b. Utilize the hardware architecture and component functions for a modern microprocessor and systems board.

c. Demonstrate usage of various memory addressing modes.

d. Explain the purpose of editors, assemblers, linkers, loaders, and operating systems runtime requirements.

e. Evaluate assemblers, compilers, and interpreters.

f. Analyze and implement memory and program segmentation.

g. Design problem solutions and utilize a processor instruction set to construct assembly language programs.

h. Design and program logic and control structures within assembly language programs.

i. Evaluate and create basic and advanced data manipulation and I/O processes within assembly language programs.

j. Implement advanced assembly language processes through MACROS, program modules, and program interrupts.

**IV. METHODS OF ASSESSMENT (TYPICAL)**

**A. FORMATIVE ASSESSMENT**

1. Assignments

2. Quizzes

3. Lab Activities

4. Exams

**B. SUMMATIVE ASSESSMENT**

1. Assignments

2. Quizzes

3. Lab Activities

4. Exams