CSIS-118B: COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE

Effective Term
Fall 2026

BOT Approval Date
05/15/2025

Discipline
CSIS - Computer Sci/Info Systems
Course Number
118B

Course Title
Computer Organization & Assembly Language

Short Title
Comp Org & Assem Lang

Credit Status (CB 04)
D - Credit - Degree applicable

Units

Lecture Units
2.00

Lab Units
1.00

Total Units
3.00

Hours

Lecture Contact Hours
32-36
Out of Class Hours Lecture
64-72

Lab Contact Hours
48-54
Out of Class Hours Lab
0.00

Total Contact Hours
80 - 90

Total Out of Class Hours
64 - 72

Total Student Learning Hours
144 - 162

Is this an existing Extensive Lab?
No

Do you want to propose this for Extensive Lab?
No

Distance Education
Yes

Distance Education

Distance Education Type
Both Fully Online and Hybrid Online

Fully Online Delivery Requirements:
a. Students must be notified via the college schedule of classes and the syllabus for the class if proctored tests are required for this course
b. Any planned face-to-face meetings, such as an orientation or study session, must be optional
c. The MSJC curriculum committee requires the use of accessible, asynchronous discussion as a component of every fully online course

Are you using publisher content?
No

Regular Substantive Interaction
Instructor-to-student, student-to-student, and student-to-content contact shall be distributed in a manner that will ensure regular substantive interaction (RSI) is maintained not only weekly but also for the duration of the course. Instructor interactions should be equivalent in time, quality and consistency to the engagement students would receive in a face-to-face course—for example, through announcements, discussion participation, personalized feedback, video messages, or other instructional communications. Instructors are expected to be responsive to student inquiries within a 24-72 hour time frame excluding holidays. RSI will be maintained through a variety of methods: • Orientation - Students must be oriented to the online and face-to-face portions at the start of the course. • Announcements - Announcements using the course management system must be posted at least weekly to keep students current on course events, due dates, materials, etc. • Discussion Forums - Content-related discussion forums within the designated CMS will include appropriate instructor participation and will be initiated and maintained, with timely feedback provided. Instructors are required to monitor the discussions and related activities to provide guidance and direction. Open-ended discussion boards may be used to further learning and interaction. • MSJC Communication – Instructor will use MSJC-administered communication tools (such as email or the designated CMS communication tools) to regularly communicate with and provide ease of access to students. • Timely Feedback - Timely feedback on student work will be provided by the instructor, which may include comments and/or rubrics, within the Grades area of the designated CMS. • Scheduled Meetings – Hybrid and synchronous courses will meet at regularly scheduled times. ADA Compliance - All course interactions and content must be designed with accessibility in mind and meet ADA requirements.

C-ID (Admin Only)
C-ID COMP 142

Catalog Description
This course is an introduction to the hardware organization and assembly language of the Intel processor. Topics include memory hierarchy and design, CPU design, pipelining, addressing modes, subroutine linkage, polled input/output, interrupts, high level language interfacing and macros.

Codes

TOP Code (CB 03)
0707.10* - *Computer Programming

CIP Code
11.0701 - Computer Science.

Repeatability Code
No

Grading Option
Letter grade OR P/NP

Minimum Qualifications

Minimum Qualifications	And/Or
Computer Science (Masters Required)	Or
Computer Information Systems	End

GE Information

Check GE Types Requested

AA/AS D2 - Communication & Analytical Thinking

Learning Objectives

Learning Objectives

Learning Objective
1. Distinguish the differences between the Von Newman and Harvard computer architectures.
2. Distinguish the differences of the general-purpose registers and their uses.
3. Compare the relationship of assembly language to high-level languages, and the processes of compilation, linking and execution cycles.
4. Develop an introductory understanding of the structure and operations of memory, virtual memory, cache, storage, and pipelining.
5. Develop an in-depth understanding of interrupt handling and exceptions.
6. Evaluate the relationship of assembly language and the architecture of the machine; this includes the addressing system, how instructions and variables are stored in memory, and the fetch-and-execute cycle.
7. Construct basic assembly language programs using the 80x86 architecture.
8. Evaluate potential biases in algorithms, data sets, and software systems that may impact diverse populations.
9. Develop inclusive coding practices to ensure fairness and equity in software design.

Learning Outcome Information

Course Learning Outcomes

Learning Outcome
1. Students will demonstrate an understanding of the data representations including Number bases, Signed numbers, and two's compliment representation.
2. Students will display an understanding of the x86 architecture, Microcomputer design, and the von Neumann machine
3. Students will compose assembly programs that handle data transfers as well as indirect and direct addressing.
4. Students will demonstrate the use of the stack and how it relates to function calls.

Content

Course Lecture Content

1. Hardware and Software Architecture
   a. Components of a Microcomputer
   b. System Architecture
   c. System Software and Memory
2. Assembly Language Fundamentals
   a. Data Definition Directives
   b. Data Transfer Instructions
   c. Arithmetic Instructions
   d. Addressing Modes
   e. The Macro Assembler
3. The Assembly Process
   a. File Relations
   b. Equates
   c. Operators and Expressions

   d. Control Transfer Instructions
   e. Debugging
   f. Input Output Services
   g. Procedures
   h. Software Interrupts
   i. DOS Function Calls
   j. BIOS Level Video Control
4. Conditional Processing
   a. Boolean and Comparison Instructions
   b. Conditional Jumps
   c. Conditional Loops
   d. High Level Logic Structures
5. Arithmetic Instructions
   a. Shift and Rotate Instructions
   b. Multiple
   c. Addition and Subtraction
   d. Signed Arithmetic
   e. Multiplication and Division
   f. ASCII Arithmetic
   g. Packed Decimal Arithmetic
6. Number Conversions and Libraries
   a. Character translation
   b. Using XLAT
   c. Binary To ASCII conversion
   d. ASCII To Binary Conversion
7. Separately Assembled Files
   a. Creating External Subroutines
   b. Stack Parameters
8. String Processing
   a. String Storage Methods
   b. String Primitive Instructions
   c. Creating a Library of string routines
   d. Creating a Link Library
9. Macros and Structures
   a. Declaring and calling macros
   b. Passing Parameters
   c. Nested Macros
   d. Macros Calling Procedures
   e. Conditional Assembly with Macros
   f. Macro Operations
   g. Defining Repeat Blocks
10. Type Operators
   a. STRUC Directive
   b. RECORD Directive
11. Disk Storage
   a. Physical and Logical Characteristics
   b. Types of Disks
   c. Disk Formats
   d. Directory Format
   e. File Allocations Table (FAT 32) NTFS
   f. System Level File Disk Functions
   g. Drive and Directory Manipulation
12. File Manipulation

   a. File Processing Standards
   b. File Functions
   c. Random Access Files
13. High Level Linking
   a. Linking to C and C++
   b. Inline statements and directives
14. System Hardware
   a. Real Time Clock
   b. CPU
15. Calculating Instruction
   a. Timings Dynamic Memory Allocation
   b. Modify Memory Blocks
   c. Allocate Memory Release
   d. Allocated Memory
16. Hardware Interrupt Handling
   a. Replacing Interrupt Vectors
   b. Hardware Control
   c. Using I/O Ports
17. Diversity, Equity, and Inclusion
   a. Recognize and Mitigate Bias in Programming
   b. Impact of Technology on Diverse Communities
   c. Accessibility Standards
   d. Ethical Responsibilities of Programmers

Course Lab Content

1. Assembly Language Fundamentals
   a. Data Definition Directives
   b. Data Transfer Instructions
   c. Arithmetic Instructions
   d. The Macro Assembler
2. The assembly Process
   a. Equates Operators and Expressions
   b. Control Transfer Instructions
   c. Debugging
   d. Input Output Services
   e. Procedures
   f. Software Interrupts
3. Conditional Processing
   a. Boolean and Comparison Instructions
   b. Conditional Jumps
4. Arithmetic Instructions
   a. Multiple Addition and Subtraction
   b. Signed Arithmetic
   c. Multiplication and Division
5. String Processing
   a. Creating a Library of string routines
   b. Creating a Link Library
6. Macros and Structures
   a. Passing Parameters
   b. Macros
   c. Calling Procedures
7. Type Operators
   a. STRUC Directive
   b. RECORD Directive
8. Disk Storage

a. System Level File-Disk Functions
b. Drive and Directory Manipulation

9. File Manipulation
   a. File Processing
   b. Standard File Functions
   c. Random Access Files
10. High-Level Linking
    a. Linking to C and C++
11. Dynamic Memory Allocation
    a. Modify Memory Blocks
    b. Allocate Memory
    c. Release Allocated Memory
12. Hardware Interrupt Handling
    a. Replacing Interrupt Vectors
    b. Hardware Control
    c. Using I/O Ports
13. Diversity, Equity, and Inclusion
    a. Recognize and Mitigate Bias in Programming
    b. Impact of Technology on Diverse Communities
    c. Accessibility Standards
    d. Ethical Responsibilities of Programmers

Methods of Instruction

Method

Discussion

Integration

Regular discussions will be conducted to evaluate potential biases in algorithms, data sets, and software systems that may impact diverse populations; compare and contrast the use and implementation of data representations, including number bases, signed numbers, and two-compliment representation; the relationship of assembly language to high-level languages; and the processes of compilation, linking, and execution cycles.

DE Adaptations for MOI

The discussion tool in the course management system will be used to conduct weekly discussions organized into structured threads. This will make it easier for students to follow and contribute relevant conversations. Any external link within the discussion thread will have descriptive text explaining the link's destination. Images included in any discussion topic will have alternate text that provides a detailed image description. Any resources or documents will comply with accessibility standards.

Method

Observation and Demonstration

Integration

Observation and demonstration will illustrate lecture principles and reading assignments focusing on fundamental assembly programming concepts and applying them to problem-solving and potential solutions to homework assignments. Guided practice will include constructing basic assembly language programs using the 80x86 architecture; The structure and operations of memory, virtual memory, cache, storage, and pipelining; and the development of inclusive code practices.

DE Adaptations for MOI

Guided practice and demonstrations will be presented via closed-captioned video with synchronized text, where students follow the instructor's lead in writing code. Hands-on activities can be similarly accommodated and supplemented with software simulations. Any external link provided will have descriptive text explaining the link's destination. All resources or documents will comply with accessibility standards. The course management system will present all interactive demonstrations and video-guided practice materials.

Method

Lecture

Integration

Lecture, with supporting visual materials (PowerPoint presentation or multimedia) will introduce conceptual and practical skills such as interrupt handling and exceptions, how instructions and variables are stored in memory, and the fetch-and-execute cycle

DE Adaptations for MOI

Lectures will be delivered using video capture software to embed voice into PowerPoint slides. Larger videos will be broken down into shorter, digestible segments. All videos will be closed-captioned with synchronized audio. Any resources or documents will comply with accessibility standards. Transcripts of the lectures will be provided upon request. All lectures will be delivered using the course management system.

Methods of Evaluation

Method

Other

Integration

Discussion and guided practice will consist of assembly syntax and logic problems. Such problems may include the processes of compilation, linking, and execution cycles; the addressing system, how instructions and variables are stored in memory; and interrupt handling and exceptions. In addition, potential biases in algorithms, data sets, and software systems will be evaluated.

Guided practice will be evaluated based on following all directions, compiling without error, the project running to completion with proper input and output, and program correctness.

DE Adaptations for MOE

Guided practice will be submitted via the course management system's assignment editor. Optional methods of delivering guided practice include using the discussion, blog, and wiki tools of the course management system. Any external link provided will have descriptive text explaining the link's destination. All resources or documents will comply with accessibility standards.

Method

Exams/Tests

Integration

A midterm and final exam, composed of short answer questions and small programming examples, will be given. The exams will evaluate the mastery of topics such as data representations, interrupt handling, exceptions, virtual memory, cache, storage, pipelining, and equity in software design.

Exams will be evaluated based on following all directions, compiling without error, code running to completion with proper input and output, and program correctness.

DE Adaptations for MOE

Exams will be submitted via the course management system's assignment editor. All exams will have alternate text provided for images that give a detailed image description. Any external link within the assignment will have descriptive text explaining the link's destination. If videos are provided, they will be closed captioned with synchronized text. Any resources or documents will comply with accessibility standards. Feedback will be provided to the students via the course management system.

Method

Homework

Integration

Weekly programming assignments that apply important concepts and skills to the student's experience outside the classroom will be given. The focus might be on the Von Newman and Harvard computer architectures, interrupt handling and exceptions, potential biases in algorithms and data sets, and the processes of compilation, linking, and execution cycles.

Homework will be evaluated based on following all directions, compiling without error, code running to completion with proper input and output, and program correctness.

DE Adaptations for MOE

Assignments will be administered and collected using the course management system's assignment tool. They will be created using accessible file formats such as HTML, accessible PDF, and text documents. Any links used within an assignment will have descriptive text explaining the link's destination. Any included videos will be closed captioned with synchronized text. Any resources or documents will comply with accessibility standards.

Feedback will be provided to students via the comments section within the assignments tool of the course management system.

Assignments

Assignment Type

Writing

In-class and/or outside of class

In-class

Formative or Summative

Formative

Assignment

This assignment aims to write a program that processes and sorts data while critically analyzing potential biases in the algorithm and datasets. Explore how even low-level operations can contribute to biased outcomes and develop strategies to mitigate these issues. Your task is to sort a given dataset of names and prioritize them for inclusion in a limited display area. Sort the names alphabetically and ensure that all names are treated equitably, regardless of length, case, or special characters (e.g., accented letters).

After implementing the algorithm, write a short analysis that addresses the following:
• How might the sorting algorithm unintentionally disadvantage certain names (e.g., handling of special characters, case sensitivity, or non-English letters)?
• Changes were made to the program to ensure a fair and accurate representation of all names.
• Broader implications of biased sorting algorithms in real-world applications (e.g., hiring systems, customer prioritization).

How assignment will be adapted in online format

This assignment will be distributed and collected through the assignments area of the course management system. Feedback on the assignment will be given in the comments section of the grade book for the assignment, which is part of the course management system.

Any external link within the assignment will have descriptive text explaining the link's destination. Any resources or documents will comply with accessibility standards.

Assignment Type

Writing

In-class and/or outside of class

Outside of class

Formative or Summative

Summative

Assignment

This assignment aims to assess your ability to write complex assembly language programs that involve advanced data manipulation, control structures, and system interactions. You will implement various algorithms, work with file I/O, and apply system-level operations.

Fibonacci Sequence with Recursive and Iterative Implementation

1. Recursive Fibonacci

Write an assembly program that calculates the Nth Fibonacci number using a recursive procedure. The user will provide the value of N.
Part B: Iterative Fibonacci

2. Write a second assembly program that calculates the Nth Fibonacci number using an iterative loop.
Requirements:

3. Implement proper procedure calls and manage the stack efficiently in the recursive version.
Compare the performance of both versions by measuring execution time (e.g., using the system clock).

How assignment will be adapted in online format

This assignment will be distributed and collected through the assignments area of the course management system. Feedback on the assignment will be given in the comments section of the grade book for the assignment, which is part of the course management system.

Any external link within the assignment will have descriptive text explaining the link's destination. Any resources or documents will comply with accessibility standards.

Assignment Type

Reading

In-class and/or outside of class

Outside of class

Formative or Summative

Formative

Assignment

This reading assignment is designed to introduce you to the fundamental concepts of assembly language programming. By the end of this assignment, you should have a solid understanding of how assembly language works, the structure of assembly instructions, and the relationship between assembly code and the underlying hardware.

Chapter 1: Introduction to Assembly Language
Source: "Assembly Language for x86 Processors" by Kip Irvine (or equivalent textbook)

Topics to focus on:
What is Assembly Language?
Relationship between assembly language and machine code.
High-level overview of CPU architecture (registers, memory, and the instruction set).
Advantages and disadvantages of using assembly language.

Write a 1-2 page summary of what you learned from the readings.

How assignment will be adapted in online format

This assignment will be distributed and collected through the assignments area of the course management system. Feedback on the assignment will be given in the comments section of the grade book for the assignment, which is part of the course management system.

Any external link within the assignment will have descriptive text explaining the link's destination. Any resources or documents will comply with accessibility standards.

Course Materials

Textbooks

Irvine, Kip (2020). Assembly Language for x86 Processors, 8th Pearson. ISBN: 9780135381656

Class Size Information

Class Size

35

Class Size Category

B - Facilitated learning 35

Diversity and Inclusion

Course Design and Materials: Course will incorporate diverse perspectives, authors, and examples that reflect a variety of cultures, identities, and lived experiences. Instructional materials will be reviewed regularly to ensure representation, inclusivity, and alignment with equity-minded practices across disciplines.

Inclusive Learning Environment: Instructors will establish classroom norms and policies that promote respect, civil discourse, and belonging in both online and face-to-face classes. Clear expectations for communication and engagement will support a learning space where all students feel valued and heard.

Accessible Instruction: Course content will be delivered using accessible design. Materials will follow accessibility guidelines within the CMS, including captioned media, readable document formats, and multiple means of engagement to support diverse learning.

Equitable Teaching Practices: Instructors will use transparent and grading criteria, varied assessment methods, and timely feedback to promote fairness and student success. Opportunities for revision, reflection, and multiple demonstrations of learning may be incorporated when appropriate to the discipline.

Culturally Responsive Pedagogies: Instructional strategies will encourage critical thinking about diverse perspectives and systems of power relevant to the course content. Faculty will foster opportunities for students to connect course material to real-world contexts.

Ongoing Reflection and Improvement: Faculty will engage in continuous professional development related to IDEAA principles and reflect on course practices using metacognition to identify opportunities for increased equity, inclusion, and accessibility.

Explain how diversity and inclusion(e.g., gender, culture/race, sexuality, etc.) is infused into the course.

This course will highlight diverse groups' contributions to computer science and assembly language development while emphasizing the importance of developing inclusive coding practices that ensure fairness and equity in software design. For example, the course discussions will examine Ada Lovelace's pioneering work and Grace Hopper's transformative contributions, illustrating how individuals from varied backgrounds have shaped the field. Additionally, the lectures will feature real-world case studies, such as the development of assistive technologies that demonstrate how inclusive design can lead to accessible and equitable software solutions for marginalized communities.

Describe how assignments, instruction, evaluation, course content, and interactions are contextualized for diverse learners promoting an equitable course.

This course will offer a mix of hands-on coding tasks, written reflections, and collaborative projects. It will emphasize developing inclusive coding practices to ensure fairness and equity in software design. While some students may excel in coding, others might perform better in assignments that require problem-solving explanations or conceptual discussions.

Additionally, the course will incorporate homework assignments that focus on culturally relevant problems for a diverse range of students. These assignments will include real-world scenarios applying assembly language in contexts meaningful to different communities, further reinforcing the importance of inclusive coding practices in promoting equitable software design.

Interactions

Explain how the course will incorporate student-to-student interaction.

Incorporating collaboration, peer support, and knowledge-sharing will achieve student-to-student interaction. An example might be code review sessions where students review each other's code, provide constructive feedback, and suggest improvements.
This will promote critical thinking and attention to detail, encourage students to learn from other's mistakes and successes, and build a sense of community by fostering mutual support. I addition, the use of the discussion board will promote peer-to-peer problem solving, and collaborative learning. The discussion board will also be used as a tool where students can get conceptual understanding and clarification from each other.

Explain how the course will incorporate instructor-to-student interaction.

Instructor-to-student interaction is vital in an assembly language course, as it helps students build confidence, provides guidance, and facilitates a more profound understanding of complex concepts.
This course will achieve this by providing detailed, individualized feedback on assignments, exams, and projects to help students understand their strengths and areas for improvement.
This course will also offer regular office hours and opportunities for students to receive individual help outside of class. In addition, interactive lectures will include live demonstrations, question-and-answer sessions, and real-time coding exercises. These will help students connect theoretical knowledge with practical implementation and provide immediate feedback and clarification of concepts.

Explain how the course will incorporate student-to-content interaction.

Student-to-content interaction is a critical aspect of learning, especially in a course as technical as assembly language. These interactions engage students directly with the material, helping them practice and apply what they learn.

In this course, student-to-content interaction will be achieved through hands-on coding exercises. These exercises allow students to directly engage with assembly language concepts by writing and running code. The course will also provide interactive tutorials and simulations that help students explore concepts step-by-step at their own pace. In addition, the course will include debugging exercises, which will help students develop critical thinking and troubleshooting skills by engaging them in finding and fixing errors in assembly code.

Explain how the course will incorporate student-to-self interaction (metacognitive).

This course will promote student-to-self interactions by providing self-assessment activities so students can gauge their own understanding and identify areas for improvement. The course content will have several self-evaluation quizzes to gauge students' understanding of the course material. The self-evaluation quizzes will offer immediate feedback to help students identify topics they have mastered and those that need further study.

Key: 1043