

About this course

Computer Organization and Assembly Language Programming deals with lower level computer programming—machine or assembly language, and how these are used in the typical computer system. An assembler can translate a program from assembly language into a loader code for loading into the computer's memory for execution.

Specific topics covered include:

- Program Execution
- Performance
- Data Representation
- Instructions
- Assembly Programming
- Procedures in Assembly
- CPU Design Choices
- Memory
- Multiprocessing

Required prior knowledge and skills

Familiarity with:

- Basic C/C++ or Java Programming
- Digital Logic Concepts (Signals, Boolean logic, memory, etc)

Learning Outcomes

Learners completing this course will be able to:

- Explain how programs written in high-level languages are executed by a computer system.
- Explain what hardware factors impact program performance and how to write programs for performance.
- Explain data representation, instruction sets and addressing modes.
- Write assembly language programs employing flow control constructs and procedures.
- Explain techniques used by computer hardware designers to improve performance.
- Explain how a data path can be implemented as a single cycle or pipelined design.
- Explain how the memory hierarchy impacts performance.
- Explain the reason for the ongoing transition to multiprocessor architectures.

Estimated Workload/Time Commitment Per Week

- About 5 hours per module
- 30 to 60 minutes per quiz
- 150 minutes for Final Exam

Textbook

Most course materials will be provided as videos and readings in the course pages themselves. Throughout the course you will see recommended readings from the recommended textbook, Computer Organization and Design, Fifth Edition, Patterson & Hennessy (ISBN-13: 978-0124077263), but this textbook is not required to be successful in the course.

Course Completion and Outline

You will need to pass the final exam with an 80% or better, and successfully pass proctoring requirements for the final exam.

Module 1: Course Plan

- 1.0 : Course Plan Overview
- Pre-Test

Module 2: Program Execution

- 2.0: Overview
- 2.0: Program Execution
- 2.1: Inside the Compiler
- 2.1: Inside the Compiler
- 2.1: Inside the Compiler: Knowledge Check
- 2.2: Inside a CPU
- 2.2: Inside a CPU: Knowledge Check
- 2.3: Executing a Program
- 2.3: Executing a Program: Knowledge Check

Module 3: Performance

- 3.0: Overview
- 3.0: Performance
- 3.1: Measuring Computer Performance
- 3.1: Measuring Computer Performance: Knowledge Check
- 3.2: Propagation Delay
- 3.2: Propagation Delay: Knowledge Check
- 3.3: Clock Rate
- 3.3: Clock Rate: Knowledge Check
- 3.4: Execution Time
- 3.4: Execution Time: Knowledge Check

Module 4: Data Representation

- 4.0: Overview
- 4.0: Data Representation
- 4.1: Binary Number System
- 4.1 Binary Number System: Knowledge Check
- 4.2: Hexadecimal Number System
- 4.2 Hexadecimal Number System: Knowledge Check
- 4.3: Negatives in Binary
- 4.3 2's Complement: Knowledge Check
- 4.4: Binary Arithmetic
- 4.4 Binary Arithmetic: Knowledge Check

- 4.5: Bounds of Binary
- 4.5 Overflow and Underflow: Knowledge Check
- 4.6: Data in Memory
- 4.6 Endianness: Knowledge Check

Module 5: Instructions

- 5.0: Overview
- 5.0: Instructions
- 5.1: Instruction Format
- 5.1 Instruction Format and Addressing Modes: Knowledge Check
- 5.2: Assembling Instructions
- 5.2 Assembling Instructions: Knowledge Check

Module 6 - Assembly Programming

- 6.0: Overview
- 6.0: MIPS Assembly Programming
- 6.1: Memory and Registers
- 6.1 Memory and Registers: Knowledge Check
- 6.2: Arithmetic Operations
- 6.2 Arithmetic Operations: Knowledge Check
- 6.3: Branching
- 6.3 Branching: Knowledge Check
- 6.4: Comparisons
- 6.4 Comparisons: Knowledge Check
- 6.5: Loops
- 6.5 Loops: Knowledge Check

Module 7: Procedures in Assembly

- 7.0: Overview
- 7.0: Procedures in Assembly
- 7.1: Inside a Procedure Call
- 7.1 Procedure Call Process: Knowledge Check
- 7.2: Nested Procedures and the Stack
- 7.2 Nested Procedures: Knowledge Check
- 7.3: Procedure Example

Module 8: CPU Design Choices

- 8.0: Overview
- 8.0: CPU Design Choices
- 8.1: Pipelining
- 8.1 Pipelining: Knowledge Check
- 8.2: Pipeline Hazards
- 8.2 Pipeline Hazards: Knowledge Check
- 8.3: Branch Prediction
- 8.3 Branch Prediction: Knowledge Check

Module 9: Memory

- 9.0: Overview
- 9.0: Memory
- 9.1: Memory Technologies
- 9.1 Memory Technologies: Knowledge Check
- 9.2: Memory Hierarchy
- 9.2 Memory Hierarchy: Knowledge Check

Module 10: Multiprocessing

- 10.0: Overview
- 10.0: Multiprocessing
- 10.1: Multiprocessing Architectures
- 10.1 Multiprocessing: Knowledge Check

Course Completion

- Final Exam Proctoring Setup
- Final Exam

Creators



Ryan Meuth, Lecturer

In 2013, Ryan Meuth joined Arizona State University in the School of Computing, Informatics, and Decision System Engineering as a lecturer, focusing on Freshman engineering and the first year experience. Since then, he has additionally become the course coordinator for the CSE Capstone Program, ASU101-CSE, and FSE100-CSE. Meuth teaches FSE100-CSE, CSE Capstone, and a variety of CSE program courses focusing on embedded systems. Meuth was awarded the Top 5% Teaching Award in 2014, 2015 and 2016, as well as the Spirit of EPICS award in Spring 2015. Meuth's research interests are focused on the computer science classroom experience, and how that can be improved for all students.



Phillip Miller, Lecturer

Phill Miller joined the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University in 2014 after teaching computer science and software engineering courses for 14 years at the University of Advancing Technology in Tempe, AZ.