Catalog Description

This is the first course of the CS2001-2002 course sequence. The objective of this course sequence is to teach foundational CS concepts by constructing a general-purpose computer system from the ground up. By integrating computer architecture, compilers, operating systems, and algorithms through hands-on projects, students will explore hardware and software design techniques and balance tradeoffs between competing design constraints. Students will learn about the engineering of systems across many levels of abstraction, from digital logic to software design. In this course, computing abstractions are introduced from first principles, taking logical behaviors of circuit components and combining them through recursive ascent to provide modular units of logical behavior. Co-requisites: CS2020 and MA2025 (may be enrolled concurrently).

Program Learning Objectives

This course sequence presents foundational knowledge in a Computer Science degree program necessary for understanding topics in more specialized courses. The course does not individually or directly satisfy ESRs or degree requirements, but its fundamental nature lays an important part of the foundation for all ESR and degree requirement courses.

Learning Outcomes

Upon successfully completing the course sequence, the student will be able to:
• Understand foundational principles of computer science, including computer architecture, compilers, operating systems, and algorithms, by constructing a general-purpose computer system from the ground up.
• Balance tradeoffs between key design constraints.
• Apply the von Neumann Model and the concepts of cache memory, virtual memory, memory segmentation, paging, and address translation to the construction of a general-purpose computer system.
• Apply the principle of abstraction to solve computer system design problems.
• Accelerate programs by identifying hardware and software bottlenecks and understanding the interface between software and physical devices.

Upon successfully completing the course, the student will be able to:
• Understand how processors are designed and organized.
• Characterize how a general-purpose processor and its associated memory hierarchy unite to execute instructions and perform functions.
• Apply Boolean logic to the design of digital circuits.
• Construct complex systems from primitive digital circuits.
• Translate programs from assembly language to machine code.
• Design circuits in a simple Hardware Description Language (HDL).
Assessment

To assess the learning outcomes, students will demonstrate their knowledge on a sequence of tests. Students will also perform laboratory exercises, and lab reports are part of the assessment. The projects are described in the textbook and are supported by machine simulation software provided by the authors. Running code the students have translated on the simulated machines provides immediate feedback on correctness and basic troubleshooting information in cases of incorrect translations.

Assessments for Course Sequence Learning Outcomes:
• Understand foundational principles of Computer Science: Labs 1-6 and Tests
• Balancing of tradeoffs: Labs 1-6 and Tests
• Apply von Neumann Model, etc.: Labs 5-6 and Test on material from Weeks 10 and 11: "A Closer Look at ISAs"
• Apply abstraction to solve design problems: Labs 1-6 and Tests
• Accelerate programs: Labs 4 and 6

Assessments for Course Learning Outcomes:
• Understand how processors are designed and organized: Labs 1-5
• Characterizing the organization of a processor and the memory hierarchy: Labs 3-5 and Test on material from Weeks 8 and 9: "Basic Machines and Instruction Execution"
• Apply Boolean logic: Labs 1-5 and Test on Week 3 material: "Basic Logic Circuits"
• Construct complex systems from primitive digital circuits: Labs 1-5
• Translate programs from assembly language to machine code: Lab 6
• Design circuits in a simple Hardware Description Language (HDL): Labs 1-5

Credit hours

This is a 3-3 class. The course will meet for six hours per week, organized as three lecture hours and three lab hours.

Co-requisites

CS2020 and MA2025 (may be enrolled concurrently).

Textbook

## Topics Covered

<table>
<thead>
<tr>
<th>Weeks</th>
<th>Topic</th>
<th>Subtopics</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>How Machines Compute</td>
<td>Demonstration of system to be built, abstraction/implementation paradigm, cost metrics, historical development, models of program execution, Moore’s Law</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 1: Elementary Logic Gates</em></td>
</tr>
<tr>
<td>2</td>
<td>Data Representation</td>
<td>Conversion between bases, signed integer representation, floating-point representation, character codes, error detection and correction</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 2: Combinational Chips</em></td>
</tr>
<tr>
<td>1.5</td>
<td>Basic Logic Circuits</td>
<td>Boolean function evaluation, Boolean algebra for circuit descriptions, logic gates</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 3: Sequential Chips</em></td>
</tr>
<tr>
<td>1.5</td>
<td>Hardware Organization</td>
<td>Instruction decoding, functional units, clock strategies, race conditions, interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 4: Machine Language Programming</em></td>
</tr>
<tr>
<td>2</td>
<td>Basic Machines and Instruction Execution</td>
<td>Instruction set, basic CPU operations, assembly programming, register and stack organization, fetch-decode-execute cycle</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 5: Computer Architecture</em></td>
</tr>
<tr>
<td>2</td>
<td>A Closer Look at Instruction Set Architectures (ISAs)</td>
<td>Instruction formats, instruction types, addressing, real-world examples of ISAs</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Project 6: The Assembler</em></td>
</tr>
</tbody>
</table>