The Hebrew University of Jerusalem

Syllabus

Computer Cons. Workshop: From Nand to Tetris - 67925

Last update 10-03-2025

HU Credits: 5

Degree/Cycle: 1st degree (Bachelor)

Responsible Department: Computer Sciences

Academic year: 0

Semester: 1st and/or 2nd Semester

Teaching Languages: English

Campus: E. Safra

Course/Module Coordinator: Mr. Yitzchak Vaknin

Coordinator Email: nand2tet@gmail.com

Coordinator Office Hours: See Moodle page

Teaching Staff:
Mr. Yitzchak Vaknin

Course/Module description:
In this course, students will build a complete computer system — both hardware and software — from the ground up. The work is organized around 12 projects, each corresponding to a chapter in the accompanying textbook.

We will begin with NAND gates and D flip-flops, and from these basic building blocks, construct all the hardware components of a simple yet complete computer. Students will then program this computer using an assembly language, for which they will write an assembler.
Next, they will develop a compiler for a simple high-level, object-based language, using a stack-based virtual machine (VM) as an intermediate level.

Finally, students will create a software library that functions as a basic operating system, and on top of it, develop a simple computer game.

All this will be accomplished within a single semester, thanks to the extreme simplicity of each component, the simulators and testing tools provided, and the carefully designed course structure.

By the end of the course, students will have a comprehensive understanding of how a complete computer system works — an understanding that will serve as a foundation for many topics in computer science.

The course is based on independent work, guided by precise specifications. Weekly meetings are dedicated to clarifications, broader context, and practical insights. Work may be done individually or in pairs, and the final grade is based both on ongoing assignments and a final evaluation.

The course is open to second-year undergraduate students and above.

Course Roadmap:

Chapter 0: Under the Hood
An introductory chapter that presents a demo game (Pong) running on the simulated computer, to motivate the entire project. It raises essential questions: what hardware and software components are needed to make this application possible? This sets the stage for a full top-down overview of the system we are about to build.

Chapter 1: Boolean Logic
Starting from NAND gates, students build basic logic gates (AND, OR, NOT, XOR),

multiplexers, and multi-bit versions. Chips are implemented and tested using HDL (Hardware Description Language) and a hardware simulator.

Chapter 2: Boolean Arithmetic
Students implement basic arithmetic units: half-adder, full-adder, and multi-bit adders. Finally, they design and implement an Arithmetic Logic Unit (ALU).

Chapter 3: Sequential Logic
Using D flip-flops, students build sequential memory elements: registers, RAM units, and a counter that serves as the program counter.

Chapter 4: Machine Language
Introduction to the Hack machine language — a simple assembly language designed for the computer architecture built in this course.

Chapter 5: Computer Architecture
Integration of all previous components into a working hardware platform, including the CPU, memory, and I/O. Students explore how the architecture runs machine-language programs and how it interfaces with a memory-mapped screen and keyboard.

Chapter 6: Assembler
Students implement an assembler that translates Hack assembly code into binary machine code. The assembler is typically implemented in Python, but other object-oriented languages are acceptable.

Chapters 7–8: Virtual Machine
Design and implementation of a stack-based virtual machine that will serve as the compiler's intermediate target.
Chapter 7: Stack arithmetic and logical operations.
Chapter 8: Procedure calls, including call stack management.

Chapter 9: High-Level Programming (Jack Language)
Introduction to Jack, a simple, Java-like, object-based programming language. Students write interactive programs and games (e.g., Pong, Tetris).

Chapters 10–11: Compilation
Development of a two-phase Jack compiler:
Chapter 10: Syntax analysis (parsing).
Chapter 11: Code generation (translating Jack to VM code).

Chapters 10-11: Compilation. We specify a Jack compiler designed to translate a collection of Jack classes into VM code. Chapter 10 focuses on syntax analysis, and Chapter 11 on semantics and code generation.

Chapter 12: Operating System

Development of a basic OS library providing essential services: math utilities, string handling, I/O, and memory management.

Course/Module aims:

- Students will build basic electronic components of a computer.
- Students will implement three compilers for languages at different abstraction levels.
- Students will develop a basic operating system.

Learning outcomes - On successful completion of this module, students should be able to:

See "Course Aims" section.

Attendance requirements(%):

0

Teaching arrangement and method of instruction: Independent work.
Throughout the course, students complete 12 projects that follow the chapters of the course book.

Course/Module Content:

1.Boolean Logic.
2.Boolean Arithmetic.
3.Sequential Logic.
4.Machine Language.
5.Computer Architecture.
6.Assembler.
7.Virtual Machine (Arithmetic).
8.Continue Virtual Machine (Control)
9.High Level Programming.
10.Compilation.
11.Continue Compilation (Code Generation)
12.Operating System.

Required Reading:

Nisan, Noam, and Schocken, Shimon. The Elements of Computing Systems, Second Edition: Building a Modern Computer from First Principles. Cambridge: MIT Press, 2021.

Additional Reading Material:

NA

Grading Scheme:

Computerized Exam - Personal Computer % 60

Submission assignments during the semester: Exercises / Essays / Audits / Reports / Forum / Simulation / others 40 %

Additional information:

This course is an independent workshop, with most of the materials provided in English.

*A final exam in *Hebrew* will be held on campus if possible.*
The exam will take place via Exam Moodle (BYOD – Bring Your Own Device) and may require additional tools as determined by the university, such as SEB.

The course requires prior knowledge of Python, particularly for text processing tasks.