האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

The Hebrew University of Jerusalem

Syllabus

COMPUTER CONS. WORKSHOP: FROM NAND TO TETRIS - 67925

Last update 09-01-2014

HU Credits: *5*

Degree/Cycle: *1st degree (Bachelor)*

Responsible Department: *Computer Science*

Academic year: *2*

Semester: *2nd Semester*

Teaching Languages: *Hebrew*

Campus: *E. Safra*

Course/Module Coordinator: *Uri Heinemann*

Coordinator Email: *Uri.Heinemann@mail.huji.ac.il*

Coordinator Office Hours: *11:00-12:00 Tuesday*

Teaching Staff:
Uri Heinemann

Course/Module description:
This course leads the students thourgh the process of building a complete computer system (hardware and software). This course consists of 11 projects, each based on a chapter in the accompanying book. Each chapter cosists of the specification and hints for implementation of a well-defined sub-system that can be built and tested in isolation. The first four chapters focus on constructing the hardware platform of a simple modern computer. Five chapters build higher level programming possibilities for this hardware, starting from an assembler and ending with a compiler for a high level object-based programming language. Two chapters are focused on code written in that programming language: a mini operating system, and a game application. The complete game plan is as follows:

Importantly, each chapter/project is a stand-alone unit that includes a complete description of its interface with neighboring chapters.
Chapter 0: Under the hood. We illustrate a sample application - the Pong game - running on the simulated computer. This motivates the question of what is needed in order to make this application a reality. We need a programming language, an operating system, a hierarchy of software translators, and a suitable hardware platform. This sets the stage to a top-down overview of the book plan and the construction projects that lie ahead.
Chapter 1: Boolean Logic. Beginning at the Nand level, we specify and build a set of elementary logic gates, multiplexors, and their multi-bit versions. In addition, we describe how chips can be built and tested using a simple version of HDL (Hardware Description Language) and the supplied Hardware Simulator.
Chapter 2: Boolean Arithmetic. Continuing with combinational logic, we specify and build a set of adders. half-adder, full-adder, and parallel adder. Next, we specify an Arithmetic Logic Unit (ALU) and describe its proposed implementation.
Chapter 3: Sequential Logic. Beginning with D-Flip-Flops, we build a ecursive hierarchy of memory systems: 1-bit register, multi-bit register, and a Random Access Memory (RAM). We also build a counter chip, which will later function as the computer's program counter.
Chapter 4: Computer Architecture. We specify a simple machine language, giving both its binary and symbolic instruction sets. Next, we guide the students through the process of integrating all the previously built chips into a unified architecture, capable of executing programs written in the specified language. We also specify how the architecture interacts with memory-mapped screen and keyboard devices. This completes the construction of the hardware platform.
Chapter 5: Assembler. Following an overview of the machine and assembly languages presented in the previous chapter, we specify an assembler. We expect the students to implement this assembler (as well as all the subsequent translators) in Java, but other object-oriented languages can also be used.

Chapters 6-7: Virtual Machine. We discuss the virtues of a virtual machine approach, and specify a stack-based VM. Next, we guide the students through the process of writing a VM implementation, which will later serve as the backend of the compiler. Chapter 6 focuses on stack arithmetic, and chapter 7 on the procedure call stack.
Chapter 8: High Level Programming. We present a simple, Java-like object-based language, called Jack. The students then write several simple Jack programs, e.g. games like Pong and Tetris. These programs are highly interactive and event-driven.
Chapters 9-10: Compilation. We specify a Jack compiler, designed to translate a collection of Jack classes into VM code. Chapter 9 focuses on syntax analysis and chapter 10 on semantics and code generation.
Chapter 11: Operating System. We use several programming examples to motivate the need for an operating system (OS). We then specify the API of a simple OS that includes math functions, string processing, input/output functionality, and memory management.

Course/Module aims:
Building a basic electronic components.
Programming three compilers for languages at different levels.
Building a basic operating system.

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

Attendance requirements(%):
0

Teaching arrangement and method of instruction: Self-practice.
Students carry 12 projects corresponding to the chapters in 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:
The Element of Computing System
By Noam Nisan and Shimon Schocken.

Additional Reading Material:
NA

Course/Module evaluation:
End of year written/oral examination 30 %
Presentation 0 %
Participation in Tutorials 0 %
Project work 0 %
Assignments 70 %
Reports 0 %
Research project 0 %
Quizzes 0 %
Other 0 %

Additional information:
NA