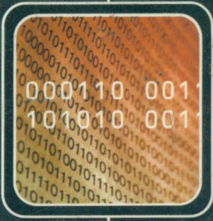
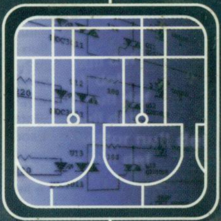


# introduction to computing systems

## from bits & gates to C & beyond

yale n. patt
sanjay j. patel



F9X

004
P215

# INTRODUCTION TO COMPUTING SYSTEMS:

*From Bits and Gates to C and Beyond*

**Yale N. Patt**
*The University of Texas at Austin*

**Sanjay J. Patel**
*University of Illinois at Urbana-Champaign*

McGraw Hill

Boston Burr Ridge, IL Dubuque, IA Madison, WI
New York San Francisco St. Louis
Bangkok Bogotá Caracas Lisbon London Madrid Mexico City
Milan New Delhi Seoul Singapore Sydney Taipei Toronto

# PREFACE

This textbook has evolved from EECS 100, the first computing course for computer science, computer engineering, and electrical engineering majors at the University of Michigan, that Kevin Compton and the first author introduced for the first time in the fall term, 1995.

EECS 100 happened because Computer Science and Engineering faculty had been dissatisfied for many years with the lack of student comprehension of some very basic concepts. For example, students had a lot of trouble with pointer variables. Recursion seemed to be "magic," beyond understanding.

We decided in 1993 that the conventional wisdom of starting with a high-level programming language, which was the way we (and most universities) were doing it, had its shortcomings. We decided that the reason students were not getting it was that they were forced to memorize technical details when they did not understand the basic underpinnings.

The result is the bottom-up approach taken in this book. We treat (in order) MOS transistors (very briefly, long enough for students to grasp their global switch-level behavior), logic gates, latches, logic structures (MUX, Decoder, Adder, gated latches), finally culminating in an implementation of memory. From there, we move on to the Von Neumann model of execution, then a simple computer (the LC-2), machine language programming of the LC-2, assembly language programming of the LC-2, the high level language C, recursion, pointers, arrays, and finally some elementary data structures.

We do not endorse today's popular information hiding approach when it comes to learning. Information hiding is a useful productivity enhancement technique after one understands what is going on. But until one gets to that point, we insist that information hiding gets in the way of understanding. Thus, we continually build on what has gone before, so that nothing is magic, and everything can be tied to the foundation that has already been laid.

We should point out that we do not disagree with the notion of top-down *design*. On the contrary, we believe strongly that top-down design is correct design. But there is a clear difference between how one approaches a design problem (after one understands the underlying building blocks), and what it takes to get to the point where one does understand the building blocks. In short, we believe in top-down design, but bottom-up learning for understanding.

## WHAT IS IN THE BOOK

The book breaks down into two major segments, a) the underlying structure of a computer, as manifested in the LC-2; and b) programming in a high level language, in our case C.

### The LC-2

We start with the underpinnings that are needed to understand the workings of a real computer. Chapter 2 introduces the bit and arithmetic and logical operations on bits. Then we begin to build the structure needed to understand the LC-2. Chapter 3 takes the student from a MOS transistor, step by step, to a real memory. Our real memory consists of 4 words of 3 bits each, rather than 64 megabytes. The picture fits on a single page (Figure 3.20), making it easy for a student to grasp. By the time the students get there, they have been exposed to all the elements that make memory work. Chapter 4 introduces the Von Neumann execution model, as a lead-in to Chapter 5, the LC-2.

The LC-2 is a 16-bit architecture that includes physical I/O via keyboard and monitor; TRAPs to the operating system for handling service calls; conditional branches on N, Z, and P condition codes; a subroutine call/return mechanism; a minimal set of operate instructions (ADD, AND, and NOT); and various addressing modes for loads and stores (direct, indirect, base+offset, and an immediate mode for loading effective addresses).

Chapter 6 is devoted to programming methodology (stepwise refinement) and debugging, and Chapter 7 is an introduction to assembly language programming. We have developed a simulator and an assembler for the LC-2. Actually, we have developed two simulators, one that runs on Windows platforms and one that runs on UNIX. The Windows simulator is available on the website and on the CD-ROM. Students who would rather use the UNIX version can download and install the software from the web at no charge.

Students use the simulator to test and debug programs written in LC-2 machine language and in LC-2 assembly language. The simulator allows on-line debugging (deposit, examine, single-step, set breakpoint, and so on). The simulator can be used for simple LC-2 machine language and assembly language programming assignments, which are essential for students to master the concepts presented throughout the first 10 chapters.

Assembly language is taught, but not to train expert assembly language programmers. Indeed, if the purpose was to train assembly language programmers, the material would be presented in an upper-level course, not in an introductory course for freshmen. Rather, the material is presented in Chapter 7 because it is consistent with the paradigm of the book. In our bottom-up approach, by the time the student reaches Chapter 7, he/she can handle the process of transforming assembly language programs to sequences of 0s and 1s. We go through the process of assembly step-by-step for a very simple LC-2 Assembler. By hand assembling, the student (at a very small additional cost in time) reinforces the important fundamental concept of translation.

It is also the case that assembly language provides a user-friendly notation to describe machine instructions, something that is particularly useful for the second half of the book. Starting in Chapter 11, when we teach the semantics of C statements, it is far easier for the reader to deal with ADD R1, R2, R3 than with 0001001010000011.

Chapter 8 deals with physical input (from a keyboard) and output (to a monitor). Chapter 9 deals with TRAPs to the operating system, and subroutine calls and returns. Students study the operating system routines (written in LC-2 code) for carrying out physical I/O invoked by the TRAP instruction.

The first half of the book concludes with Chapter 10, a treatment of stacks and data conversion at the LC-2 level, and a comprehensive example that makes use of both. The example is the simulation of a calculator, which is implemented by a main program and 11 subroutines.

## The Language C

From there, we move on to C. The C programming language occupies the second half of the book. By the time the student gets to C, he/she has an understanding of the layers below.

The C programming language fits very nicely with our bottom-up approach. Its low-level nature allows students to see clearly the connection between software and the underlying hardware. In this book we focus on basic concepts such as control structures, functions, and arrays. Once basic programming concepts are mastered, it is a short step for students to learn more advanced concepts such as objects and abstraction.

# CONTENTS

*Thông tin tài trợ dự án!*

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................

................................................................................................................................................