From Basics to Bytecode:
A Guide to Computers
and Programming

Set Lonnert
& ChatGPT 4.o from OpenAI[a]

[a] https://openai.com/

Contents

Introduction **i**

1 Foundations **1**

1.1 Simple data types . . . 1

1.1.1 Integers in binary . . . 2

1.1.2 Floating-point numbers . . . 3

1.1.3 Characters and ASCII . . . 3

1.1.4 Strings . . . 4

1.1.5 Representations and types . . . 4

1.1.6 Summary . . . 5

1.2 Variables . . . 6

1.2.1 Assignment . . . 6

1.2.2 Mutable and immutable variables . . . 8

1.2.3 Summary . . . 10

1.3 Control structures . . . 11

1.3.1 Conventional control structures . . . 12

1.3.2 Control structures in computers . . . 15

1.3.3 Summary . . . 16

1.4 Functions . . . 17

1.4.1 Calling functions . . . 18

1.4.2 Summary . . . 21

1.5 Practice . . . 22

2 Understanding VMs **25**

2.1 Simple VMs . . . 25

2.1.1 The stack . . . 26

2.1.2 Interpreter technique . . . 27

2.1.3 VM1 implementation . . . 27

2.1.4 REGVM implementation . . . 28

2.1.5 Portability . . . 32

2.1.6 Summary . . . 32

2.2 VM2 implementation . . . 34

2.2.1 Comparisons . . . 36

2.2.2 Error handling . . . 38

2.2.3 Summary 39
2.3 VM3 implementation 40
2.3.1 Frame pointer 44
2.3.2 Local storage 46
2.3.3 Memory management 47
2.3.4 Frame stack 48
2.3.5 Summary 52
2.4 Practice 54

3 Debugging, Optimisation, and Tests 55
3.1 Prerequisites 55
3.2 Basic development tools 55
3.3 Debugging 56
3.3.1 Introduction to Debugging 56
3.3.2 Debugging Strategies 56
3.3.3 Common Debugging Tools 56
3.4 Optimization 56
3.4.1 Introduction to Code Optimization 56
3.4.2 Profiling and Benchmarking 57
3.4.3 Code Optimization Techniques 57
3.5 Testing 57
3.5.1 Introduction to Software Testing 57
3.5.2 Unit Testing 57
3.5.3 Integration Testing 57
3.5.4 Performance Testing 58
3.5.5 Automated Testing and Continuous Integration 58
3.5.6 Debugging with Tests (Test-Driven Development) 58
3.5.7 Debugging 58
3.5.8 Tools 59
3.5.9 Optimisation 65
3.5.10 Memory 70
3.5.11 Time 71
3.5.12 Summary 73
3.6 Practice 74

4 Building and Experimenting 77
4.1 The computer as hardware 77
4.1.1 Hardware 79
4.1.2 The Pico 81
4.1.3 Summary 84
4.1.4 Practice 85
4.2 Input/Output 85
4.2.1 The Pico pins 86
4.2.2 Light switching circuit 87
4.2.3 Programmable I/O 89

4.2.4 State machines 91
4.2.5 Circuit for traffic lights 94
4.2.6 Pedestrian crossing 95
4.2.7 Temperature measurements 97
4.2.8 Temperature indicator 98
4.2.9 Summary 100
4.2.10 Practice 102
4.3 Secondary memory 105
4.3.1 Save and load temperatures 107
4.3.2 Summary 107
4.3.3 Practice 107
4.4 External communication 107
4.4.1 UART 108
4.4.2 Connect Two Picos 110
4.4.3 Client-Server 112
4.4.4 Wireless 115
4.5 Security 115
4.5.1 Summary 115
4.5.2 Practice 116
4.6 Displays 117
4.6.1 Text 118
4.6.2 Pixel graphics 118
4.6.3 Vector graphics 118
4.6.4 Summary 118
4.6.5 Practice 118
4.7 Combining I/O 118
4.7.1 Simplified web server 118
4.7.2 Summary 118
4.7.3 Practice 118
4.8 Handling errors in Pico 118
4.8.1 Types 119
4.8.2 Approaches 119
4.8.3 Summary 122
4.9 Timing and timers 122
4.9.1 Timing and Timers in the Pico 123
4.9.2 Interrupt handling 124
4.9.3 Summary 124
4.9.4 Practice 124
4.10 Concurrency and multithreading 125
4.11 Power management 129
4.12 Practice 133

5 Compilers 135
5.1 Parsing and parsers 135
5.2 Compilers 135

Introduction

This book is an experiment. It is an exploratory book, designed to guide readers from already acquired fundamentals of programming to complex real-world applications, with an emphasis on learning by doing. It builds upward from basic concepts, encouraging readers to engage deeply with both small-scale and large-scale programming environments. And it assumes using Large Language Models (LLMs), which assist you with feedback, code suggestions, and deeper understanding.

Part 1: Foundations of Programming. We begin with an introduction to fundamental programming concepts, including variables, control structures, and functions. Along the way, we'll explore these topics in greater depth and cover related foundational concepts. To enhance your understanding, you'll be encouraged to use LLMs alongside your learning process. These tools will offer immediate feedback, code suggestions, and explanations, helping you grasp the material more effectively.

Part 2: Understanding Virtual Machines. Transition from basic programming into the concept of virtual machines. We will use simple examples of virtual machines, and gradually introduce more complex virtual machines, linking each one to programming concepts previously covered.

Part 3: Building and Experimenting with Virtual Machines. Here we encourage you to modify and experiment with the provided virtual machine examples. This part includes project ideas for readers to explore. Integrate LLMs to help you debug, optimise, and understand changes. Use exercises at the end of each chapter that ask students to extend the virtual machines or apply them to solve specific problems.

Part 4: Compilers and Advanced Concepts. Introduce compilers and parsing with practical examples. Show how the virtual machines introduced earlier can be used to run compiled code. Provide hands-on projects where students build simple compilers or interpreters for their own small languages.