

Contents

| | |
|---|-----------|
| Introduction | i |
| 1 Foundations | 1 |
| 1.1 Prerequisites | 1 |
| 1.2 Foundations of Programming | 1 |
| 1.3 Simple Data Types | 1 |
| 1.3.1 Integers in Binary | 2 |
| 1.3.2 Floating-point Numbers | 3 |
| 1.3.3 Characters and ASCII | 4 |
| 1.3.4 Strings | 4 |
| 1.3.5 Representations and Types | 4 |
| 1.3.6 Type Systems and Type Checking | 5 |
| 1.3.7 Summary | 5 |
| 1.4 Variables | 6 |
| 1.4.1 Assignment | 6 |
| 1.4.2 Mutable and Immutable Variables | 7 |
| 1.4.3 Summary | 10 |
| 1.5 Control Structures | 11 |
| 1.5.1 Conventional Control Structures | 11 |
| 1.5.2 Control Structures in Computers | 12 |
| 1.5.3 Summary | 15 |
| 1.6 Functions | 16 |
| 1.6.1 Calling Functions | 16 |
| 1.6.2 Summary | 18 |
| 1.7 Practice | 19 |
| 2 Understanding VMs | 21 |
| 2.1 Prerequisites | 21 |
| 2.2 Simple VMs | 21 |
| 2.2.1 The Stack | 22 |

| | | |
|----------|--|-----------|
| 2.2.2 | Interpreter technique | 23 |
| 2.2.3 | VM1 Implementation | 24 |
| 2.2.4 | REGVM Implementation | 25 |
| 2.2.5 | Portability | 28 |
| 2.2.6 | Summary | 30 |
| 2.3 | Stack-based VM | 31 |
| 2.3.1 | Comparisons | 33 |
| 2.3.2 | Iterations | 34 |
| 2.3.3 | Error Handling | 35 |
| 2.3.4 | Summary | 38 |
| 2.4 | Memory and Functions | 38 |
| 2.4.1 | Frame Pointer | 42 |
| 2.4.2 | Local Storage | 44 |
| 2.4.3 | Memory Management | 45 |
| 2.4.4 | Frame stack | 46 |
| 2.4.5 | Summary | 51 |
| 2.5 | Practice | 53 |
| 3 | Development Environment | 57 |
| 3.1 | Prerequisites | 57 |
| 3.2 | Basic tools | 57 |
| 3.3 | Debugging | 58 |
| 3.3.1 | Process | 60 |
| 3.3.2 | Root Cause Analysis | 62 |
| 3.3.3 | Tools | 65 |
| 3.3.4 | Debugging Distributed and Concurrent Systems | 72 |
| 3.3.5 | Summary | 78 |
| 3.4 | Optimisation | 78 |
| 3.4.1 | Memory | 83 |
| 3.4.2 | Performance | 85 |
| 3.4.3 | Dynamic Programming | 87 |
| 3.4.4 | Confusion Matrix | 89 |
| 3.4.5 | Summary | 95 |
| 3.5 | Tests and Testing | 96 |
| 3.5.1 | Automated Testing and Continuous Integration | 101 |
| 3.5.2 | A Custom Test Virtual Machine | 104 |
| 3.5.3 | Summary | 105 |
| 3.6 | Advanced Debugging Practices | 106 |
| 3.6.1 | Error Handling and Resilience | 106 |
| 3.6.2 | Production and Live System Debugging | 112 |
| 3.6.3 | Summary | 125 |

| | | |
|----------|---|------------|
| 3.7 | The Impact of LLMs | 125 |
| 3.7.1 | Debugging | 126 |
| 3.7.2 | Optimisation | 126 |
| 3.7.3 | Testing | 127 |
| 3.7.4 | Broader Perspectives and Challenges | 128 |
| 3.7.5 | Summary | 128 |
| 3.8 | Practice | 129 |
| 4 | Building and Experimenting | 133 |
| 4.1 | Prerequisites | 133 |
| 4.2 | The Computer as Hardware | 133 |
| 4.2.1 | Hardware | 135 |
| 4.2.2 | The Pico | 137 |
| 4.2.3 | Summary | 140 |
| 4.2.4 | Practice | 141 |
| 4.3 | Basic Input/Output | 143 |
| 4.3.1 | The Pico Pins | 144 |
| 4.3.2 | Light Switching Circuit | 145 |
| 4.3.3 | Programmable I/O | 147 |
| 4.3.4 | State Machines | 149 |
| 4.3.5 | Circuit for Traffic Lights | 152 |
| 4.3.6 | Pedestrian Crossing | 154 |
| 4.3.7 | Temperature Measurements | 156 |
| 4.3.8 | Temperature Indicator | 157 |
| 4.3.9 | Summary | 162 |
| 4.3.10 | Practice | 164 |
| 4.4 | Secondary Memory | 165 |
| 4.4.1 | Save and Load Temperatures | 168 |
| 4.4.2 | A Simple Database on SD Card | 169 |
| 4.4.3 | Summary | 172 |
| 4.4.4 | Practice | 172 |
| 4.5 | External Communication | 173 |
| 4.5.1 | Universal Asynchronous Receiver-Transmitter | 173 |
| 4.5.2 | Inter-Pico Communication | 178 |
| 4.5.3 | Client-Server Architecture | 182 |
| 4.5.4 | Wireless Communication | 188 |
| 4.5.5 | WiFi Communication Between Pico W Boards | 190 |
| 4.5.6 | Advanced WiFi Features | 195 |
| 4.5.7 | Features | 198 |
| 4.5.8 | Network Protocol Stack Integration | 199 |
| 4.5.9 | Socket Programming | 200 |

| | | |
|--------|---|-----|
| 4.5.10 | Summary | 205 |
| 4.5.11 | Practice | 206 |
| 4.6 | Cryptography and Secure Communication | 207 |
| 4.6.1 | Core Principles | 207 |
| 4.6.2 | Symmetric vs Asymmetric Encryption | 207 |
| 4.6.3 | Symmetric Encryption | 208 |
| 4.6.4 | Asymmetric Encryption | 209 |
| 4.6.5 | Data Integrity and Hashing | 216 |
| 4.6.6 | Hybrid Cryptographic Systems | 217 |
| 4.6.7 | Digital Signatures | 220 |
| 4.6.8 | Implementation Considerations | 220 |
| 4.6.9 | Summary | 222 |
| 4.6.10 | Practice | 222 |
| 4.7 | Graphics | 224 |
| 4.7.1 | Framebuffers and DMA | 224 |
| 4.7.2 | Raster and Vector Approaches | 227 |
| 4.7.3 | Display Interfaces | 231 |
| 4.7.4 | Pimoroni Pico Display Pack 2.0 | 232 |
| 4.7.5 | Colour Models and Bit Depth | 235 |
| 4.7.6 | Rendering Strategies | 236 |
| 4.7.7 | Simple Double Buffering Example | 236 |
| 4.7.8 | Performance Considerations | 237 |
| 4.7.9 | Text Rendering | 237 |
| 4.7.10 | Special Techniques and Hardware | 238 |
| 4.7.11 | Summary | 239 |
| 4.7.12 | Practice | 240 |
| 4.8 | Handling errors | 241 |
| 4.8.1 | Error Types | 242 |
| 4.8.2 | Approaches | 243 |
| 4.8.3 | Summary | 246 |
| 4.8.4 | Practice | 246 |
| 4.9 | Timing and timers | 247 |
| 4.9.1 | Timing and Timers in the Pico | 249 |
| 4.9.2 | Interrupt handling | 250 |
| 4.9.3 | Summary | 252 |
| 4.9.4 | Practice | 252 |
| 4.10 | Concurrency and multithreading | 253 |
| 4.10.1 | Summary | 258 |
| 4.10.2 | Practice | 258 |
| 4.11 | Power management | 259 |
| 4.11.1 | Summary | 264 |

| | | |
|----------|--|------------|
| 4.11.2 | Practice | 264 |
| 4.12 | Practice | 265 |
| 5 | The Compiler Pipeline | 271 |
| 5.1 | Prerequisites | 271 |
| 5.2 | Compilers, Interpreters, Assemblers, and VMs | 271 |
| 5.3 | A Short History | 274 |
| 5.4 | The Pipeline in Principle | 275 |
| 5.5 | Syntax | 276 |
| 5.5.1 | Tokenisation or Lexical Analysis | 277 |
| 5.5.2 | Parsing | 277 |
| 5.5.3 | Production Rules and Grammars | 280 |
| 5.5.4 | Building Abstract Syntax Trees | 281 |
| 5.5.5 | Summary | 281 |
| 5.6 | Semantic Analysis | 282 |
| 5.6.1 | How Semantic Analysis Works | 282 |
| 5.6.2 | Symbol Table | 283 |
| 5.6.3 | Type System | 284 |
| 5.6.4 | Using Abstract Syntax Trees | 285 |
| 5.6.5 | Control Flow Graph | 289 |
| 5.6.6 | Example: Control Flow Graph | 293 |
| 5.6.7 | Data Flow and Dependency Analysis | 297 |
| 5.6.8 | Attribute Grammars | 298 |
| 5.6.9 | Consistency and Contract Checking | 299 |
| 5.6.10 | Summary | 300 |
| 5.7 | Intermediate Representations | 300 |
| 5.7.1 | Static Single Assignment | 302 |
| 5.7.2 | Three Address Code | 305 |
| 5.7.3 | Directed Acyclic Graphs | 306 |
| 5.7.4 | Summary | 311 |
| 5.8 | Code Generation Techniques | 311 |
| 5.8.1 | Example: Tail-Recursion | 320 |
| 5.8.2 | Summary | 324 |
| 5.9 | Loader and Linker | 325 |
| 5.9.1 | Linkers | 325 |
| 5.9.2 | Loaders | 326 |
| 5.9.3 | Real Examples | 326 |
| 5.9.4 | Simple Simulation | 327 |
| 5.9.5 | Summary | 331 |
| 5.10 | Optimisation in Compilers | 332 |
| 5.10.1 | Types of Compiler Optimisations | 333 |

| | | |
|----------|---|------------|
| 5.10.2 | The Future of Compiler Optimisation | 337 |
| 5.10.3 | Summary | 337 |
| 5.11 | Error Handling | 338 |
| 5.11.1 | Warnings | 338 |
| 5.11.2 | Errors | 339 |
| 5.11.3 | Future of Error Handling | 342 |
| 5.11.4 | Summary | 343 |
| 5.12 | Practice | 343 |
| 6 | Philosophy and Methodology | 353 |
| 6.1 | Philosophy and style | 353 |
| 6.1.1 | Programming as craft | 354 |
| 6.1.2 | Building process | 355 |
| 6.1.3 | Blending philosophies | 358 |
| 6.1.4 | Methodologies | 360 |
| 6.1.5 | References | 363 |
| 6.1.6 | A personal note | 365 |
| 6.1.7 | Risks | 367 |
| 6.1.8 | Summary | 368 |
| 6.2 | Problems and methods | 369 |
| 6.2.1 | Example: Low customer satisfaction | 369 |
| 6.2.2 | Example: Thermostat | 369 |
| 6.2.3 | Generalisations | 371 |
| 6.2.4 | Methodology in the craft philosophy | 374 |
| 6.2.5 | Summary | 377 |
| 6.3 | Reflections on The History and Future | 378 |
| 6.3.1 | AI Example: LLMs | 380 |
| 6.3.2 | Summary | 383 |
| 6.4 | Method examples | 383 |
| 6.4.1 | Mockups and prototypes | 384 |
| 6.4.2 | Code review | 388 |
| 6.4.3 | Null hypothesis | 394 |
| 6.4.4 | Null Hypothesis in Programming | 396 |
| 6.4.5 | Summary | 399 |
| 6.5 | Practice | 400 |
| 7 | The Mechanics | 403 |
| 7.1 | Data: Foundations, Structures, and Algorithms | 403 |
| 7.1.1 | Elementary Data Types | 404 |
| 7.1.2 | Data Structures | 404 |
| 7.1.3 | Abstract Data Types | 405 |

| | | |
|-------|--|-----|
| 7.1.4 | Algorithms | 406 |
| 7.1.5 | Summary | 406 |
| 7.2 | Types and Type Systems | 407 |
| 7.2.1 | Data Types | 407 |
| 7.2.2 | Type Systems | 407 |
| 7.2.3 | Type Errors | 408 |
| 7.2.4 | Static vs Dynamic Typing | 408 |
| 7.2.5 | Type Inference | 409 |
| 7.2.6 | Type Consistency and Soundness | 409 |
| 7.2.7 | Strong vs. Weak Typing | 409 |
| 7.2.8 | Beyond Data Safety | 410 |
| 7.2.9 | Summary | 410 |
| 7.3 | Programming Paradigms and Language Design | 410 |
| 7.3.1 | Imperative Languages | 412 |
| 7.3.2 | Functional Languages | 413 |
| 7.3.3 | Object-Oriented Languages | 417 |
| 7.3.4 | Concatenative Languages | 423 |
| 7.3.5 | Homoiconicity and Metaprogramming | 426 |
| 7.3.6 | Domain-Specific Languages | 426 |
| 7.3.7 | Example: Rust | 433 |
| 7.3.8 | Summary | 434 |
| 7.4 | Low-Level Constructs | 436 |
| 7.4.1 | Callback | 437 |
| 7.4.2 | State Machine | 439 |
| 7.4.3 | Checkpoint | 440 |
| 7.4.4 | Low-Level Programming Concepts | 442 |
| 7.4.5 | Summary | 444 |
| 7.5 | Architecture in Programming | 444 |
| 7.5.1 | History | 445 |
| 7.5.2 | Styles and Patterns | 445 |
| 7.5.3 | Change | 447 |
| 7.5.4 | Code Architecture vs. System Architecture | 447 |
| 7.5.5 | Practical Considerations in Modern Architectures | 448 |
| 7.5.6 | Summary | 449 |
| 7.6 | Design Patterns | 449 |
| 7.6.1 | Factory | 451 |
| 7.6.2 | Observer | 453 |
| 7.6.3 | Command | 456 |
| 7.6.4 | Strategy | 458 |
| 7.6.5 | Adapter | 461 |
| 7.6.6 | Composite | 464 |

| | | |
|----------|--|------------|
| 7.6.7 | Decorator | 466 |
| 7.6.8 | Summary | 468 |
| 7.7 | Networking and Modern Server Architectures | 468 |
| 7.7.1 | Server Systems | 470 |
| 7.7.2 | A Protocol Suite: TCP/IP | 472 |
| 7.7.3 | Web Servers | 479 |
| 7.7.4 | Application-Level Protocols and Styles | 480 |
| 7.7.5 | System-Level Architectural Patterns | 482 |
| 7.7.6 | Concurrency and Advanced Server Functions | 485 |
| 7.7.7 | Summary | 489 |
| 7.8 | Practice | 490 |
| 8 | Perspectives and Frontiers | 493 |
| 8.1 | Systemic Concepts in Computing | 493 |
| 8.1.1 | Abstraction | 493 |
| 8.1.2 | Latency | 494 |
| 8.1.3 | Resilience | 495 |
| 8.1.4 | Summary | 496 |
| 8.2 | Computer Security | 498 |
| 8.2.1 | The Modern Threat Landscape | 499 |
| 8.2.2 | Fundamental Principles | 500 |
| 8.2.3 | The Security Ecosystem | 501 |
| 8.2.4 | Human Factors | 502 |
| 8.2.5 | The Role of Cryptography | 502 |
| 8.2.6 | Principles of Cryptography | 503 |
| 8.2.7 | Cryptographic Techniques | 504 |
| 8.2.8 | Challenges in Cryptographic Security | 505 |
| 8.2.9 | Summary | 506 |
| 8.3 | Exploring the Limits of Computation | 506 |
| 8.3.1 | Summary | 512 |
| 8.4 | The Turing Machine | 513 |
| 8.4.1 | The Halting Problem | 514 |
| 8.4.2 | Proof of Undecidability | 514 |
| 8.4.3 | Visualising the Proof | 516 |
| 8.4.4 | Implications | 517 |
| 8.4.5 | Summary | 518 |
| 8.5 | Information Theory | 518 |
| 8.5.1 | Quantifying Information | 519 |
| 8.5.2 | Entropy | 519 |
| 8.5.3 | The Dance of Information | 520 |
| 8.5.4 | The Limit of Communication | 522 |

| | | |
|--------|---|-----|
| 8.5.5 | The Art of Data Compression | 523 |
| 8.5.6 | Divergence and Cross-Entropy | 524 |
| 8.5.7 | Summary | 525 |
| 8.6 | Artificial Intelligence and Machine Learning | 525 |
| 8.7 | Machine Learning | 529 |
| 8.7.1 | Three Samples | 537 |
| 8.7.2 | Summary | 549 |
| 8.8 | Deep Learning | 549 |
| 8.8.1 | Foundations | 552 |
| 8.8.2 | Training via Backpropagation | 553 |
| 8.8.3 | Convolutional Neural Networks (CNNs) | 555 |
| 8.8.4 | Recurrent Neural Networks (RNNs) and LSTMs | 556 |
| 8.8.5 | Transformers | 558 |
| 8.8.6 | Summary | 559 |
| 8.9 | Capabilities, Limitations, and Impacts of AI/ML | 559 |
| 8.9.1 | Limitations and Risks | 560 |
| 8.9.2 | Strengths and Achievements | 568 |
| 8.9.3 | Summary | 570 |
| 8.10 | AI: The Catalyst for a New Era in Programming | 571 |
| 8.10.1 | The Quality-Time Tradeoff | 572 |
| 8.10.2 | Rethinking Verification | 572 |
| 8.10.3 | Formal Approaches | 573 |
| 8.10.4 | Empirical Approaches | 573 |
| 8.10.5 | Summary | 574 |
| 8.11 | Hoare Logic | 575 |
| 8.11.1 | Inference Rules | 576 |
| 8.11.2 | Practical Examples | 578 |
| 8.11.3 | Advanced Topics | 581 |
| 8.11.4 | Proof Techniques: Loop Invariants & Failed Proofs | 582 |
| 8.11.5 | Tools and Applications | 583 |
| 8.11.6 | Limitations and Extensions | 583 |
| 8.11.7 | Summary | 584 |
| 8.12 | Property-Based Testing | 584 |
| 8.12.1 | The Idea | 585 |
| 8.12.2 | Properties vs. Examples: A Paradigm Shift | 587 |
| 8.12.3 | Foundations | 588 |
| 8.12.4 | Implementation Architecture | 593 |
| 8.12.5 | Advanced Concepts and Techniques | 598 |
| 8.12.6 | Practical Applications | 607 |
| 8.12.7 | Performance Considerations | 618 |
| 8.12.8 | Debugging and Diagnosis | 620 |

CONTENTS

| | | |
|-----------------|---|------------|
| 8.12.9 | Best Practices and Guidelines | 625 |
| 8.12.10 | Model-Based and Metamorphic Testing | 625 |
| 8.12.11 | Future Directions | 629 |
| 8.12.12 | Summary | 629 |
| 8.13 | Type Systems and Formal Reasoning | 630 |
| 8.13.1 | Lambda Calculus: A Simplified Overview | 631 |
| 8.13.2 | Dependent Types | 633 |
| 8.13.3 | Basic Concepts | 633 |
| 8.13.4 | Type Theory Basics | 634 |
| 8.13.5 | A Simple Type Theory with Dependent Types | 638 |
| 8.13.6 | Practical Applications and Examples | 640 |
| 8.13.7 | Example: Natural Deduction | 641 |
| 8.13.8 | Summary | 644 |
| 8.14 | Practice | 645 |
| Glossary | | 647 |
| Index | | 723 |