COMPUTER SYSTEMS

AN EMBEDDED APPROACH

Mc Graw Hill Education

IAN VINCE McLOUGHLIN

Computer Systems

An Embedded Approach

This page intentionally left blank

Computer Systems

An Embedded Approach

Professor Ian Vince McLoughlin

School of Computing
Medway Campus
University of Kent
Chatham, Kent
United Kingdom

Mc Graw Hill Education

New York Chicago San Francisco Athens
London Madrid Mexico City Milan
New Delhi Singapore Sydney Toronto

Copyright © 2018 by McGraw-Hill Education. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

ISBN: 978-1-26-011761-5
MHID: 1-26-011761-8

The material in this eBook also appears in the print version of this title: ISBN: 978-1-26-011760-8,
MHID: 1-26-011760-X.

eBook conversion by codeMantra
Version 1.0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions or for use in corporate training programs. To contact a representative, please visit the Contact Us page at www.mhprofessional.com.

Information contained in this work has been obtained by McGraw-Hill Education from sources believed to be reliable. However, neither McGraw-Hill Education nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw-Hill Education nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill Education and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

TERMS OF USE

This is a copyrighted work and McGraw-Hill Education and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill Education's prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED "AS IS." McGRAW-HILL EDUCATION AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill Education and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill Education nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill Education has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill Education and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

About the Author

Ian Vince McLoughlin is a professor of computing and is currently head of the School of Computing on the Medway Campus of the University of Kent, in Chatham, United Kingdom. Over a career spanning more than 30 years (so far), he has worked for industry, government, and academia on three continents, with an emphasis on research and innovation. At heart he is a computer engineer, having designed or worked on computing systems that can be found in space, flying in the troposphere, empowering the global telecommunications network, being used underwater, in daily use by emergency services, embedded within consumer devices, and helping patients speak following larynx surgery. Professor McLoughlin is proud to be a Fellow of the IET, a Senior Member of the IEEE, a Chartered Engineer (UK), and an Ingenieur European (EU).

This page intentionally left blank

Contents

Preface xvii
Acknowledgments xxi
List of Boxes xxiii

1 Introduction 1
1.1 The Evolution of Computers 1
1.2 Forward Progress 3
1.3 Computer Generations 5
1.3.1 First Generation 6
1.3.2 Second Generation 7
1.3.3 Third Generation 8
1.3.4 Fourth Generation 9
1.3.5 Fifth Generation 10
1.4 Cloud, Pervasive, Grid, and Massively Parallel Computers 12
1.5 Where To from Here? 13
1.6 Summary 16

2 Foundations 17
2.1 Computer Organization 17
2.1.1 Flynn's Taxonomy 18
2.1.2 Connection Arrangements 18
2.1.3 Layered View of Computer Organization 20
2.2 Computer Fundamentals 21
2.3 Number Formats 25
2.3.1 Unsigned Binary 26
2.3.2 Sign Magnitude 26
2.3.3 One's Complement 27
2.3.4 Two's Complement 27
2.3.5 Excess-*n* 28
2.3.6 Binary-Coded Decimal 29
2.3.7 Fractional Notation 29
2.3.8 Sign Extension 30
2.4 Arithmetic 31
2.4.1 Addition 32
2.4.2 The Parallel Carry-Propagate Adder 32
2.4.3 Carry Look-Ahead 34
2.4.4 Subtraction 35
2.5 Multiplication 37
2.5.1 Repeated Addition 38
2.5.2 Partial Products 38

2.5.3 Shift-Add Method 42
2.5.4 Booth's and Robertson's Methods 42
2.6 Division 44
2.6.1 Repeated Subtraction 44
2.7 Working with Fractional Number Formats 46
2.7.1 Arithmetic with Fractional Numbers 47
2.7.2 Multiplication and Division of Fractional Numbers 48
2.8 Floating Point 49
2.8.1 Generalized Floating Point 49
2.8.2 IEEE754 Floating Point 50
2.8.3 IEEE754 Modes 51
2.8.4 IEEE754 Number Ranges 55
2.9 Floating Point Processing 58
2.9.1 Addition and Subtraction of IEEE754 Numbers 59
2.9.2 Multiplication and Division of IEEE754 Numbers 62
2.9.3 IEEE754 Intermediate Formats 62
2.9.4 Rounding 63
2.10 Summary 64
2.11 Problems 64

3 CPU Basics 69
3.1 What Is a Computer? 69
3.2 Making the Computer Work for You 70
3.2.1 Program Storage 70
3.2.2 Memory Hierarchy 71
3.2.3 Program Transfer 73
3.2.4 Control Unit 74
3.2.5 Microcode 79
3.2.6 RISC versus CISC Approaches 81
3.2.7 Example Processor—the ARM 83
3.2.8 More about the ARM 85
3.3 Instruction Handling 86
3.3.1 The Instruction Set 86
3.3.2 Instruction Fetch and Decode 90
3.3.3 Compressed Instruction Sets 95
3.3.4 Addressing Modes 97
3.3.5 Stack Machines and Reverse Polish Notation 101
3.4 Data Handling 102
3.4.1 Data Formats and Representations 103
3.4.2 Data Flows 107
3.4.3 Data Storage 107
3.4.4 Internal Data 108
3.4.5 Data Processing 109
3.5 A Top-Down View 113
3.5.1 Computer Capabilities 113
3.5.2 Performance Measures, Statistics, and Lies 114
3.5.3 Assessing Performance 116

3.6 Summary 118
3.7 Problems 119

4 Processor Internals 123
4.1 Internal Bus Architecture 123
4.1.1 A Programmer's Perspective 123
4.1.2 Split Interconnection Arrangements 124
4.1.3 ADSP21xx Bus Arrangement 126
4.1.4 Simultaneous Data and Program Memory Access 127
4.1.5 Dual-Bus Architectures 129
4.1.6 Single-Bus Architectures 131
4.2 Arithmetic Logic Unit 132
4.2.1 ALU Functionality 132
4.2.2 ALU Design 133
4.3 Memory Management Unit 136
4.3.1 The Need for Virtual Memory 136
4.3.2 MMU Operation 136
4.3.3 Retirement Algorithms 139
4.3.4 Internal Fragmentation and Segmentation 140
4.3.5 External Fragmentation 140
4.3.6 Advanced MMUs 142
4.3.7 Memory Protection 143
4.4 Cache 144
4.4.1 Direct Cache 146
4.4.2 Set-Associative Cache 148
4.4.3 Full-Associative Caches 149
4.4.4 Locality Principles 149
4.4.5 Cache Replacement Algorithms 150
4.4.6 Cache Performance 154
4.4.7 Cache Coherency 156
4.5 Coprocessors 158
4.6 Floating Point Unit 159
4.6.1 Floating Point Emulation 160
4.7 Streaming SIMD Extensions and Multimedia Extensions 162
4.7.1 Multimedia Extensions 162
4.7.2 MMX Implementation 163
4.7.3 Use of MMX 164
4.7.4 Streaming SIMD Extensions 164
4.7.5 Using SSE and MMX 165
4.8 Coprocessing in Embedded Systems 165
4.9 Summary 167
4.10 Problems 167

5 Enhancing CPU Performance 173
5.1 Speedups 174
5.2 Pipelining 174

5.2.1 Multifunction Pipelines 176
5.2.2 Dynamic Pipelines 177
5.2.3 Changing Mode in a Pipeline 178
5.2.4 Data Dependency Hazard 180
5.2.5 Conditional Hazards 181
5.2.6 Conditional Branches 183
5.2.7 Compile-Time Pipeline Remedies 185
5.2.8 Relative Branching 187
5.2.9 Instruction Set Pipeline Remedies 188
5.2.10 Run-Time Pipeline Remedies 189
5.3 Complex and Reduced Instruction Set Computers 193
5.4 Superscalar Architectures 194
5.4.1 Simple Superscalar 194
5.4.2 Multiple-Issue Superscalar 197
5.4.3 Superscalar Performance 197
5.5 Instructions per Cycle 198
5.5.1 IPC of Difference Architectures 198
5.5.2 Measuring IPC 200
5.6 Hardware Acceleration 201
5.6.1 Zero-Overhead Loops 201
5.6.2 Address Handling Hardware 204
5.6.3 Shadow Registers 207
5.7 Branch Prediction 208
5.7.1 The Need for Branch Prediction 209
5.7.2 Single T-Bit Predictor 211
5.7.3 Two-Bit Predictor 212
5.7.4 The Counter and Shift Registers as Predictors 214
5.7.5 Local Branch Predictor 214
5.7.6 Global Branch Predictor 217
5.7.7 The Gselect Predictor 219
5.7.8 The Gshare Predictor 221
5.7.9 Hybrid Predictors 222
5.7.10 Branch Target Buffer 224
5.7.11 Basic Blocks 225
5.7.12 Branch Prediction Summary 227
5.8 Parallel and Massively Parallel Machines 227
5.8.1 Evolution of SISD to MIMD 230
5.8.2 Parallelism for Raw Performance 232
5.8.3 More on Parallel Processing 234
5.9 Tomasulo's Algorithm 239
5.9.1 The Rationale behind Tomasulo's Algorithm 239
5.9.2 An Example Tomasulo System 240
5.9.3 Tomasulo in Embedded Systems 245
5.10 Very Long Instruction Word Architectures 246
5.10.1 What Is VLIW? 246
5.10.2 The VLIW Rationale 248
5.10.3 Difficulties with VLIW 249
5.10.4 Comparison with Superscalar 250

5.11 Summary 250
5.12 Problems 251

6 Externals 255
6.1 Interfacing Using a Bus 255
6.1.1 Bus Control Signals 256
6.1.2 Direct Memory Access 257
6.2 Parallel Bus Specifications 258
6.3 Standard Interfaces 260
6.3.1 System Control Interfaces 260
6.3.2 System Data Buses 260
6.3.3 I/O Buses 267
6.3.4 Peripheral Device Buses 267
6.3.5 Interface to Networking Devices 268
6.4 Real-Time Issues 269
6.4.1 External Stimuli 269
6.4.2 Interrupts 269
6.4.3 Real-Time Definitions 270
6.4.4 Temporal Scope 270
6.4.5 Hardware Architecture Support for Real Time 272
6.5 Interrupts and Interrupt Handling 273
6.5.1 The Importance of Interrupts 274
6.5.2 The Interrupt Process 274
6.5.3 Advanced Interrupt Handling 280
6.5.4 Sharing Interrupts 280
6.5.5 Reentrant Code 281
6.5.6 Software Interrupts 281
6.6 Embedded Wireless Connectivity 282
6.6.1 Wireless Technology 282
6.6.2 Wireless Interfacing 284
6.6.3 Issues Relating to Wireless 284
6.7 Summary 285
6.8 Problems 285

7 Practical Embedded CPUs 291
7.1 Introduction 291
7.2 Microprocessors Are Core Plus More 291
7.3 Required Functionality 295
7.4 Clocking 298
7.4.1 Clock Generation 300
7.5 Clocks and Power 301
7.5.1 Propagation Delay 303
7.5.2 The Trouble with Current 303
7.5.3 Solutions for Clock Issues 304
7.5.4 Low-Power Design 304
7.6 Memory 306
7.6.1 Early Computer Memory 306

7.6.2 ROM: Read-Only Memory 307
7.6.3 RAM: Random-Access Memory 314
7.7 Pages and Overlays 321
7.8 Memory in Embedded Systems 323
7.8.1 Booting from Non-Volatile Memory 325
7.8.2 Other Memory 327
7.9 Test and Verification 328
7.9.1 IC Design and Manufacture Problems 328
7.9.2 Built-In Self-Test 331
7.9.3 JTAG 333
7.10 Error Detection and Correction 336
7.11 Watchdog Timers and Reset Supervision 340
7.11.1 Reset Supervisors and Brownout Detectors 341
7.12 Reverse Engineering 343
7.12.1 The Reverse Engineering Process 344
7.12.2 Detailed Physical Layout 348
7.13 Preventing Reverse Engineering 353
7.13.1 Passive Obfuscation of Stored Programs 355
7.13.2 Programmable Logic Families 356
7.13.3 Active RE Mitigation 357
7.13.4 Active RE Mitigation Classification 357
7.14 Soft Core Processors 358
7.14.1 Microprocessors Are More Than Cores 359
7.14.2 The Advantages of Soft Core Processors 360
7.15 Hardware Software Codesign 363
7.16 Off-the-Shelf Cores 365
7.17 Summary 367
7.18 Problems 368

8 Programming 371
8.1 Running a Program 372
8.1.1 What Does Executing Mean? 372
8.1.2 Other Things to Note 375
8.2 Writing a Program 376
8.2.1 Compiled Languages 377
8.2.2 Interpreted Languages 381
8.3 The UNIX Programming Model 383
8.3.1 The Shell 384
8.3.2 Redirections and Data Flow 385
8.3.3 Utility Software 387
8.4 Summary 388
8.5 Problems 388

9 Operating Systems 391
9.1 What Is an Operating System? 391
9.2 Why Do We Need an Operating System? 392
9.2.1 Operating System Characteristics 393
9.2.2 Types of Operating Systems 394

9.3 The Role of an Operating System 396
9.3.1 Resource Management 396
9.3.2 Virtual Machine 396
9.3.3 CPU Time 397
9.3.4 Memory Management 398
9.3.5 Storage and Filing 400
9.3.6 Protection and Error Handling 401
9.4 OS Structure 402
9.4.1 Layered Operating Systems 403
9.4.2 Client-Server Operating Systems 404
9.5 Booting 405
9.5.1 Booting from Parallel Flash 406
9.5.2 Booting from HDD/SSD 408
9.5.3 What Happens Next 409
9.6 Processes 410
9.6.1 Processes, Processors, and Concurrency 411
9.7 Scheduling 413
9.7.1 The Scheduler 414
9.8 Storage and File Systems 417
9.8.1 Secondary Storage 417
9.8.2 Need for File Systems 421
9.8.3 What Are File Systems? 424
9.8.4 Backup 431
9.9 Summary 433
9.10 Problems 434

10 Connectivity 437
10.1 Why Connect, How to Connect 437
10.1.1 One-to-One Communications 438
10.1.2 One-to-Many Communications 439
10.1.3 Packet Switching 440
10.1.4 Simple Communications Topologies 441
10.2 System Requirements 443
10.2.1 Packetization 443
10.2.2 Encoding and Decoding 445
10.2.3 Transmission 445
10.2.4 Receiving 445
10.2.5 Error Handling 446
10.2.6 Connection Management 451
10.3 Scalability, Efficiency, and Reuse 453
10.4 OSI Layers 454
10.5 Topology and Architecture 455
10.5.1 Hierarchical Network 455
10.5.2 Client-Server Architecture 456
10.5.3 Peer-to-Peer Architecture 456
10.5.4 Ad Hoc Connection 457
10.5.5 Mobility and Handoff 458

10.6 Summary 458
10.7 Problems 459

11 Networking 461
11.1 The Internet 461
11.1.1 Internet History 462
11.1.2 Internet Governance 462
11.2 TCP/IP and the IP Layer Model 464
11.2.1 Encapsulation 465
11.3 Ethernet Overview 469
11.3.1 Ethernet Data Format 470
11.3.2 Ethernet Encapsulation 471
11.3.3 Ethernet Carrier Sense 473
11.4 The Internet Layer 474
11.4.1 IP Address 474
11.4.2 Internet Packet Format 476
11.4.3 Routing 477
11.4.4 Unicasting and Multicasting 478
11.4.5 Anycasting 478
11.4.6 Naming 478
11.4.7 Domain Name Servers 479
11.5 The Transport Layer 482
11.5.1 Port Number 482
11.5.2 User Datagram Protocol 483
11.5.3 Transmission Control Protocol 483
11.5.4 UDP versus TCP 484
11.6 Other Messages 485
11.6.1 Address Resolution Protocol 485
11.6.2 Control Messages 486
11.7 Wireless Connectivity 486
11.7.1 WiFi 487
11.7.2 WiMax 487
11.7.3 Bluetooth 488
11.7.4 ZigBee 488
11.7.5 Near-Field Communications 489
11.8 Network Scales 490
11.9 Summary 490
11.10 Problems 490

12 The Future 493
12.1 Single-Bit Architectures 494
12.1.1 Bit-Serial Addition 494
12.1.2 Bit-Serial Subtraction 495
12.1.3 Bit-Serial Logic and Processing 496
12.2 More-Parallel Machines 497
12.2.1 Clusters of Small CPUs 497
12.2.2 Parallel and Cluster Processing Considerations 501
12.2.3 Interconnection Strategies 502

12.3 Asynchronous Processors 505
12.3.1 Data Flow Control 507
12.3.2 Avoiding Pipeline Hazards 508
12.4 Alternative Number Format Systems 509
12.4.1 Multiple-Valued Logic 509
12.4.2 Signed Digit Number Representation 510
12.5 Optical Computation 514
12.5.1 The Electro-Optical Full Adder 514
12.5.2 The Electro-Optic Backplane 515
12.6 Science Fiction or Future Reality? 517
12.6.1 Distributed Computing 518
12.6.2 Wetware 518
12.7 Summary 519

A Standard Memory Size Notation 521

B Standard Logic Gates 523

Index 525

This page intentionally left blank

Preface

Computers in their widest sense—including smartphones, portable gaming systems, and so on—surround us and increasingly underpin our daily lives. This book is dedicated to peeling back the layers from those systems to examine and understand what "makes them tick." That is the motivation behind the emphasis on embedded systems—that and the fact that embedded systems contain truly fascinating technology. Looking inside them, to a technically minded person, is like unwrapping a Christmas gift of knowledge and understanding.

Bookshops (particularly in university towns) seem to overflow with textbooks on topics like computer architecture, computer system design, networking, operating systems, and even embedded systems. Many famous technical authors have tried their hands at writing in this area, yet computers constitute a fluid and ever-advancing area of technology that is difficult to describe adequately with a static textbook that may rapidly become out-of-date. In particular, the rise of embedded computing systems over the past decade or so seems to have surprised some of the traditional authors: Some textbooks persist in regarding computers as being the room-sized machines of the 1950s and 1960s. Other textbooks regard computers as being primarily the desktop and server machines of the 1980s and 1990s. Only a handful of authors have truly acknowledged that the vast majority of computers in modern use are embedded within everyday objects. Few textbooks acknowledge that the future of computing is embedded, connected, and pervasive: There will come a time when the concept of a desktop or even notebook computer seems as anachronistic as the punched card machines of 50 years ago.

In *Computer Systems: An Embedded Approach*, as mentioned, we point our discussion squarely toward this embedded future wherever possible, and use examples from the embedded world, for all three subareas of computer architecture, operating systems, and connectivity. Some topics naturally relate better to embedded processors and are described in that way, but others are handled alongside the more traditional topics that dominate other texts. Wherever possible, examples are given from the embedded world, and related material introduced to describe the relevance of the topics to the readers of today.

Book Structure

The 12 chapters that constitute this textbook can, apart from the introduction and conclusion, be divided roughly into three parts that explain, in turn, the following questions:

- What hardware is inside a modern computer system (embedded or otherwise), how does it work, and how does it fit together?

- What is needed to program a computer to "do" things? How does that software get written, loaded, and executed; how is it organized and presented; and how are the systems inside a modern computer managed?
- How do computers connect together to exchange information and provide distributed servers for users?

In general, following the introduction and foundations chapter, the discussions about hardware are confined to Chapters 3 to 7, those concerning software programming and operating systems are in Chapters 8 and 9, while the networking and connectivity discussions involve Chapters 10 and 11. There is significant internal referencing between parts, but there is no reason that readers must progress through the book sequentially—the three parts contain material that can be read and understood independently of the other parts.

Target Audience

The target audience of readers includes undergraduate students of computing, computer science, computer engineering, computer systems engineering, electronic and computer engineering, electronic and electrical engineering, and variants. The book will equally appeal to those working in other technical disciplines who wish to study the foundations and fundamentals of computers and computing. The introductory sections and the foundations chapter are designed to be suitable for undergraduates in their first 2 years of study, but sufficient depth and pointers to further topics are provided to make this a suitable text for final-year students undertaking courses on computer architecture or computer systems design.

Computer programmers and engineers who are working in the embedded systems industry will find this textbook to be a useful reference, and the emphasis on the ARM processor—which is widely used across industry—will be welcome for many of those for whom this technology is now a livelihood.

Book Design

The material in this book was written from the bottom up, without being based on existing textbooks, apart from some sections in the hardware or computer architecture part of the book, which make use of sections from the author's previous work *Computer Architecture: An Embedded Approach* as a foundation. By taking a fresh approach, and planning the book without being constrained by traditional structures, the text avoids many of the historical blind alleys and irrelevant sideshows that have occurred in computer evolution. This leads to a more precisely defined flow in the writing structure that maintains the sharp focus on embedded systems (although it does not totally ignore the larger machines—many of them contained fascinating examples of ideas that morphed over the years into more important and better technology that has since became ubiquitous).

In creating this book, the author has aimed to write easy-access and readable text that builds reader interest and tries to maintain relevance with the popular forefront of technology as it impacts daily life. However, there are tricky concepts in any field of study, and where those have been encountered, care has been taken to write clear explanatory text, and in many cases provide an informative and intuitive illustration to aid understanding. In addition, there are many explanatory boxes provided throughout, which contain material such as extra worked examples, interesting snippets of information, and additional explanations. All of those features aim to augment the main text and assist the reader in absorbing the information.

SI (System International) units are used throughout the book, including the unusual-sounding "kibibyte" and "mebibyte" capacity measures for computer memory (which are explained in Appendix A). Each of the main chapters in the book is followed with end-of-chapter problems, which have answers available to instructors online. An accompanying website, www.mcloughlin.eu/computer, provides students and other readers additional reference material, links, and opportunities to find out more about many of the topics presented here.

Book Preparation

This book has been prepared and typeset with LaTeX using **TeXShop**. The author wrote the content using **TeXstudio** on Linux Ubuntu– and Apple OS-X–based computers. Line diagrams were all drawn using the OpenOffice/LibreOffice drawing tools, and all graphics conversions have made use of the extensive graphics processing tools that are freely available on GNU/Linux systems. Code examples and most of the embedded system descriptions were sourced from the author's own hardware and software designs. Thanks are gratefully expressed to the GNU Project for the excellent and invaluable GCC ARM compiler, as well as to the Busybox (the Swiss Army knife of embedded systems coding) and ARM/Linux projects.

Several of the images in this book were supplied by online resources such as Wikimedia Commons, as credited in the figure captions. All such figures are under Creative Commons (CC) by attribution (BY) or share alike (SA) licenses. The author would like to acknowledge in particular the excellent resource provided by Wikimedia Commons, as well as the protective licenses from Creative Commons.[1]

Before You Begin

Please take a moment to remember the generations of computer engineers who have worked hard for decades to bring us the mobile, smart, computer, and embedded technologies that modern society thrives on. While we can rightly applaud those great efforts of the past, it is the author's hope that readers will enjoy unwrapping the gifts of understanding and knowledge of embedded computer systems, and will work just as hard to build a better technology future for us all.

Ian Vince McLoughlin

[1]The full license text for all CC images used in this book can be viewed at https://creativecommons.org/licenses.

This page intentionally left blank

Acknowledgments

Thanks are due most of all to my patient wife Kwai Yoke and children Wesley and Vanessa, all of whose patience and encouragement have contributed to this book. More practically I would like to thank all of the editorial and production staff at McGraw-Hill Education in New York, whose encouragement and enthusiasm have been much appreciated, and whose professionalism has been key in getting this published. But I also acknowledge a debt of gratitude to Gerald Bok and others at McGraw-Hill Asia in Singapore, in particular Gerald's support of my writing career from 2006 to 2013.

I have many friends who I could (and probably should) thank here for their support, encouragement, and influence; however, I would simply like to dedicate this work to my mother. I acknowledge her constant encouragement, not just for writing this and other books, but throughout my entire lifetime. Her high expectations led to my entering academia, and she was always enthusiastic about anything related to education. I can truly say that her memory lives on in all that I do and accomplish. But above all I give glory to the God who made me, guided me, refined me, gave His son to save me, and will eventually welcome me into His presence. All that I am, accomplish, obtain, and achieve, I ultimately dedicate to Him. Except the errors (in this book or elsewhere); those are all mine.

This page intentionally left blank

List of Boxes

2.1 Worked Endiness Example 1 23
2.2 Worked Endiness Example 2 23
2.3 Worked Endiness Example 3 24
2.4 Worked Endiness Example 4 24
2.5 What Is a Number Format? 25
2.6 Negative Two's Complement Numbers 27
2.7 Worked Examples of Number Conversion 27
2.8 Is Binary a Fractional Number Format? 29
2.9 Fractional Format Worked Example 30
2.10 Sign Extension Worked Example 31
2.11 Exercise for the Reader 33
2.12 Worked Example 33
2.13 Exercise for the Reader 34
2.14 Worked Examples of Two's Complement Multiplication 40
2.15 Exercise for the Reader 43
2.16 Booth's Method Worked Example 43
2.17 Long Division Worked Example 45
2.18 Worked Examples of Fractional Representation 47
2.19 Worked Example of Fractional Division 49
2.20 IEEE754 Normalized Mode Worked Example 1 52
2.21 IEEE754 Normalized Mode Worked Example 2 52
2.22 Exercise for the Reader 53
2.23 IEEE754 Denormalized Mode Worked Example 54
2.24 IEEE754 Infinity and Other "Numbers" 54
2.25 Worked Example Converting from Decimal to Floating Point 57
2.26 Floating Point Arithmetic Worked Example 60
2.27 IEEE754 Arithmetic Worked Example 61

3.1 How the ARM Was Designed 83
3.2 Illustrating Conditionals and the S Bit in the ARM 89

3.3 Condition Codes in the ARM Processor 91
3.4 Understanding the MOV Instruction in the ARM 94
3.5 A Huffman Coding Illustration 95
3.6 Recoding RPN Instructions to Minimize Stack Space 102
3.7 Data Types in Embedded Systems 104
3.8 Standardized Performance 115

4.1 Exploring ALU Propagation Delays 134
4.2 MMU Worked Example 137
4.3 Trapping Software Errors in the C Programming Language 143
4.4 Cache Example: The Intel Pentium Pro 145
4.5 Direct Cache Example 147
4.6 Set-Associative Cache Example 148
4.7 Cache Replacement Algorithm Worked Example 1 152
4.8 Cache Replacement Algorithm Worked Example 2 153
4.9 Access Efficiency Example 155
4.10 MESI Protocol Worked Example 158
4.11 An Alternative Approach—FPU on the Early ARM Processors 160

5.1 Pipeline Speedup 176
5.2 WAW Hazard 181
5.3 Conditional Flags 182
5.4 Branch Prediction 185
5.5 Speculative Execution 186
5.6 Relative Branching 187
5.7 Scoreboarding 196
5.8 ZOL Worked Examples 205
5.9 Address Generation in the ARM 207
5.10 Aliasing in Local Prediction 217

6.1 DMA in a Commercial Processor 257
6.2 Bus Settings for Peripheral Connectivity 259
6.3 The Trouble with ISA 262
6.4 Scheduling Priorities 272
6.5 ARM Interrupt Timing Calculation 277
6.6 Memory Remapping during Boot 279

7.1 Configurable I/O Pins on the MSP430 295
7.2 Pin Control on the MSP430 296
7.3 NAND and NOR Flash Memory 310
7.4 Memory Map in the MSP430 326
7.5 Using JTAG for Finding a Soldering Fault 333
7.6 Using JTAG for Booting a CPU 335
7.7 Hamming (7, 4) Encoding Example 339
7.8 Hamming (7, 4) Encoding Example Using Matrices 340
7.9 Bus Line Pin Swapping 352

9.1 More about the Kernel 402
9.2 Finding Out about Processes 411
9.3 Practical File Systems—ext4 422
9.4 File System Configuration—ext4 423

10.1 Everyday Errors 448

11.1 RFCs 463
11.2 IPv4 Address Allocation 475

12.1 Some Examples of Large-Scale Cluster Computers 504
12.2 Example CSD Number 513

This page intentionally left blank

Computer Systems

An Embedded Approach

This page intentionally left blank