Wrox: Programmer to Programmer

Professional Embedded ARM Development

James A. Langbridge

Join the discussion at p2p.wrox.com

wrox™
A Wiley Brand

PROFESSIONAL **EMBEDDED ARM DEVELOPMENT**

INTRODUCTION xxi

➤ **PART I** **ARM SYSTEMS AND DEVELOPMENT**

CHAPTER 1 The History of ARM 3

CHAPTER 2 ARM Embedded Systems. 13

CHAPTER 3 ARM Architecture. 29

CHAPTER 4 ARM Assembly Language 53

CHAPTER 5 First Steps. 73

CHAPTER 6 Thumb Instruction Set 107

CHAPTER 7 Assembly Instructions 121

CHAPTER 8 NEON 145

CHAPTER 9 Debugging 159

CHAPTER 10 Writing Optimized C 175

➤ **PART II** **REFERENCE**

APPENDIX A Terminology 193

APPENDIX B ARM Architecture Versions 199

APPENDIX C ARM Core Versions 205

APPENDIX D NEON Intrinsics and Instructions. 215

APPENDIX E Assembly Instructions 221

INDEX 247

PROFESSIONAL

Embedded ARM Development

PROFESSIONAL

Embedded ARM Development

James A. Langbridge

wrox™
A Wiley Brand

Professional Embedded ARM Development

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
`www.wiley.com`

ISBN: 978-1-118-78894-3
ISBN: 978-1-118-78901-8 (ebk)
ISBN: 978-1-118-88782-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

ACQUISITIONS EDITOR
Mary James

PROJECT EDITOR
Christina Haviland

TECHNICAL EDITORS
Jean-Michel Hautbois
Chris Shore

PRODUCTION EDITOR
Christine Mugnolo

COPY EDITOR
San Dee Phillips

EDITORIAL MANAGER
Mary Beth Wakefield

FREELANCER EDITORIAL MANAGER
Rosemarie Graham

ASSOCIATE DIRECTOR OF MARKETING
David Mayhew

MARKETING MANAGER
Ashley Zurcher

BUSINESS MANAGER
Amy Knies

VICE PRESIDENT AND EXECUTIVE GROUP PUBLISHER
Richard Swadley

ASSOCIATE PUBLISHER
Jim Minatel

PROJECT COORDINATOR, COVER
Patrick Redmond

PROOFREADER
Nancy Carrasco

TECHNICAL PROOFREADER
Stephan Cadene

INDEXER
Robert Swanson

COVER DESIGNER
Ryan Sneed

COVER IMAGE
Background: PhotoAlto Images/Fotosearch

For my loving girlfriend, Anne-Laure, who put up with entire weekends spent on my PC (while she spent her weekend on her laptop, sending me encouraging electronic messages). Thank you for supporting me when I should have been paying attention to you.

For my wonderful daughter, Eléna. Thank you for letting daddy work when I really should have spent more time playing with you, and despite what I might have said at the time, thank you for unplugging my computer when I ignored you for too long. Your smiles and first words are what powered me through the late nights and tight deadlines.

ABOUT THE AUTHOR

JAMES A. LANGBRIDGE does not like talking about himself in the third person, but he will try anyway. James was born in Singapore, and followed his parents to several countries before settling down in Nantes, France, where he lives with his partner and their daughter.

James is an embedded systems consultant and has worked for more than 15 years on industrial, military, mobile telephony, and aviation security systems. He works primarily on low-level development, creating bootloaders or optimizing routines in assembly, making the most of small processors. When not on contract, James trains engineers on embedded systems, or he makes new gizmos, much to the dismay of his partner.

James wrote his first computer program at age six and has never stopped tinkering since. He began using Apple IIs, ZX80s and ZX81s, before moving on to BBC Micros and the Amiga, before finally having no other option but to use PCs.

ABOUT THE TECHNICAL EDITORS

CHRIS SHORE is the Training and Education Manager at ARM Ltd, based in Cambridge, UK. He has been responsible for training ARM's global customer base for over 13 years, delivering nearly 200 training courses per year on everything from chip design to software optimization. Chris has taught classes on every continent except Antarctica — opportunities there are limited, but surely it's only a matter of time! He is a regular speaker at industry conferences.

Following graduation with his degree in Computer Science from Cambridge University, Chris worked as a software consultant for over 15 years, primarily in embedded real-time systems, before moving to ARM in 1999. He is a Chartered Engineer and Member of the Institute of Engineering and Technology, and he sits on the Industry Advisory Board of Queen Mary College, London. In his free time he keeps bees, tries to play the guitar, and is always looking for ways to visit new countries.

JEAN-MICHEL HAUTBOIS lives in France and has been developing software professionally, or as a hobbyist, for more than 15 years. He is currently employed as an embedded Linux consultant with Vodalys, and is the architect of his company's main video product which was developed on an ARM-based SoC. He is involved in the decision-making process when a new hardware product needs to be created and performance is critical. In his free time Jean-Michel likes to travel, and he enjoys spending time with his wife and newborn son.

ACKNOWLEDGMENTS

I CAN'T FIND THE WORDS to thank everyone who has helped me with this project. It all started with the questions of junior developers; I hope I've answered your questions. The LinkedIn ARM group has been an excellent source of information, both by the questions asked and the quality of the answers. I would like to thank everyone at ARM who has helped me. My thanks especially to my two technical editors, Chris Shore and Jean-Michel Hautbois; without your help, this book would not have been possible. My thanks also to Philippe Lançon for your support, to Atmel and Silicon Labs for your boards and your time, and to all my friends and family who helped me with this project. And, of course, the Wiley team, who helped me every time I had a question or a doubt, and who put up with me. To Christina Haviland, Mary James, San Dee Phillips, and everyone else who helped me — thank you.

CONTENTS

INTRODUCTION *xxi*

PART I: ARM SYSTEMS AND DEVELOPMENT

CHAPTER 1: THE HISTORY OF ARM 3

The Origin of ARM 3
- Why Acorn Decided to Create a New Processor 5
- Why Acorn Became ARM 5
- Why ARM Doesn't Actually Produce Microprocessors 6

ARM Naming Conventions 7
- How to Tell What Processor You Are Dealing With 8
- Differences between ARM7TDMI and ARM926EJ-S 9
- Differences between ARM7 and ARMv7 10
- Differences between Cortex-M and Cortex-A 11

Manufacturer Documentation 11

What Is ARM Doing Today? 11

Summary 12

CHAPTER 2: ARM EMBEDDED SYSTEMS 13

ARM Embedded Systems Defined 15
- What Is a System on Chip? 15
- What's the Difference between Embedded Systems and System Programming? 16
- Why Is Optimization So Important? 17
- What Is the Advantage of a RISC Architecture? 19

Choosing the Right Processor 21

What Should You Start With? 22
- What Boards Are Available? 23
- What Operating Systems Exist? 24
- Which Compiler Is Best Suited to My Purpose? 25
- Getting Ready for Debugging 26
- Are There Any Complete Development Environments? 26
- Is There Anything Else I Need to Know? 27

Summary 28

CHAPTER 3: ARM ARCHITECTURE 29

Understanding the Basics 29
Register 30
Stack 31
Internal RAM 31
Cache 31
Getting to Know the Different ARM Subsystems 33
Presenting the Processor Registers 33
Presenting the CPSR 35
Calculation Unit 37
Pipeline 37
Tightly Coupled Memory 39
Coprocessors 39
Understanding the Different Concepts 40
What Is an Exception? 40
Handling Different Exceptions 42
Modes of Operation 43
Vector Table 44
Memory Management 45
Presenting Different Technologies 47
JTAG Debug (D) 47
Enhanced DSP (E) 47
Vector Floating Point (F) 48
EmbeddedICE (I) 48
Jazelle (J) 48
Long Multiply (M) 48
Thumb (T) 49
Synthesizable (S) 49
TrustZone 49
NEON 50
big.LITTLE 50
Summary 51

CHAPTER 4: ARM ASSEMBLY LANGUAGE 53

Introduction to Assembly Language 53
Talking to a Computer 54
Why Learn Assembly? 55
Speed 55
Size 56
Fun! 57
Compilers Aren’t Perfect 57

Understanding Computer Science through Assembly 58
Shouldn't You Just Write in Assembly? 58
Uses of Assembly 59
Writing Bootloaders 59
Reverse Engineering 59
Optimization 60
ARM Assembly Language 60
Layout 61
Instruction Format 61
Condition Codes 62
Updating Condition Flags 65
Addressing Modes 66
ARM Assembly Primer 69
Loading and Storing 69
Setting Values 69
Branching 69
Mathematics 70
Understanding an Example Program 70
Summary 71

CHAPTER 5: FIRST STEPS 73

Hello World! 74
Taking the World Apart 77
Hello World, for Real This Time! 79
Software Implementation 81
Memory Mapping 83
Real World Examples 85
Silicon Labs STK3800 85
Silicon Labs STK3200 89
Atmel D20 Xplained Pro 95
Case Study: U-Boot 102
Machine Study: Raspberry Pi 103
Boot Procedure 103
Compiling Programs for the Raspberry Pi 104
What's Next? 105
Summary 105

CHAPTER 6: THUMB INSTRUCTION SET 107

Thumb 108
Thumb-2 Technology 109
How Thumb Is Executed 109

Advantages of Using Thumb 110
Cores Using Thumb 111
ARM-Thumb Interworking 113
Introduction to Thumb-1 113
Register Availability 114
Removed Instructions 114
No Conditionals 114
Set Flags 114
No Barrel Shifter 115
Reduced Immediates 115
Stack Operations 115
Introduction to Thumb-2 115
New Instructions 116
Coprocessor 117
DSP 118
FPU 118
Writing for Thumb 118
Summary 119

CHAPTER 7: ASSEMBLY INSTRUCTIONS 121

Movement 122
MOV 122
MVN 122
MOVW 123
MOVT 123
NEG 123
Example: Loading a 32-Bit Constant from the Instruction Stream 123
Arithmetic 125
ADD 125
ADC 126
SUB 126
SBC 126
RSB 126
RSC 127
Example: Basic Math 127
Saturating Arithmetic 127
QADD 128
QSUB 128
QDADD 128
QDSUB 129

Data Transfer 129
LDR 129
STR 130
Example: memcpy 130
Logical 130
AND 131
EOR 131
ORR 131
BIC 131
CLZ 131
Compare 131
CMP 132
CMN 132
TST 132
TEQ 132
Branch 132
B 133
BL 133
BX 133
BLX 134
Example: Counting to Zero 134
Example: Thumb Interworking 134
What Is MOV pc, lr? 135
Multiply 135
MUL 135
MLA 135
UMULL 136
UMLAL 136
SMULL 136
SMLAL 136
Divide 136
SDIV 137
UDIV 137
Multiple Register Data Transfer 137
STM 138
LDM 139
Barrel Shifter 139
LSL 139
LSR 140
ASR 140

ROR 140
RRX 140
Stack Operations 140
PUSH 141
POP 141
Example: Returning from a Subroutine 141
Coprocessor Instructions 141
MRC 141
MCR 142
Miscellaneous Instructions 142
SVC 142
NOP 143
MRS 143
MSR 143
Summary 143

CHAPTER 8: NEON 145

What Are the Advantages to NEON? 145
What Data Types Does NEON Support? 147
Using NEON in Assembly 147
Presenting the Registers 147
Loading and Storing Data 148
Optimized memcpy 152
NEON Instructions 152
Using NEON in C 153
Presenting Intrinsics 154
Using NEON Intrinsics 155
Converting an Image to Grayscale 156
Summary 158

CHAPTER 9: DEBUGGING 159

What Is a Debugger? 159
What Can a Debugger Do? 160
ARM Debugging Capabilities 160
Types of Debugging 162
Loops 162
Routines 163
Interrupt Controllers 163
Bootloaders 163

Debuggers 163
GNU Debugger 163
J-Link GDB Debugger 165
Example Debugging 165
Infinite Loop 165
Unknown Exception 167
Dividing by Zero 168
In-Depth Analysis 169
Data Abort 169
Corrupted Serial Line 170
64-Bit Calculations 172
A Timely Response 173
Summary 174

CHAPTER 10: WRITING OPTIMIZED C 175

Rules for Optimized Code 175
Don't Start with Optimization 176
Know Your Compiler 176
Know Your Code 176
Profiling 176
Profiling Inside an Operating System 177
Profiling on a Bare Metal System 178
C Optimizations 180
Basic Example 180
Count Down, Not Up 182
Integers 183
Division 183
Don't Use Too Many Parameters 184
Pointers, Not Objects 185
Don't Frequently Update System Memory 185
Alignment 185
Assembly Optimizations 186
Specialized Routines 186
Handling Interrupts 186
Hardware Configuration Optimizations 187
Frequency Scaling 187
Configuring Cache 188
Summary 190

PART II: REFERENCE

APPENDIX A: TERMINOLOGY 193

Branch Prediction 193
Cache 193
Coprocessors 194
CP10 195
CP11 195
CP14 195
CP15 195
Cycle 195
Exception 195
Interrupt 196
Jazelle 196
JTAG 196
MIPS 196
NEON 196
Out-of-Order Execution 197
Pipeline 197
Register 197
SIMD 197
SOC 197
Synthesizable 197
TrustZone 198
Vector Tables 198

APPENDIX B: ARM ARCHITECTURE VERSIONS 199

ARMv1 200
ARMv2 200
ARMv3 200
ARMv4 201
ARMv5 201
ARMv6 202
ARMv6-M 202
ARMv7-A/R 203
ARMv7-M 203
ARMv8 203

APPENDIX C: ARM CORE VERSIONS 205

ARM6 205
ARM7 205

ARM7TDMI 206
ARM8 206
StrongARM 206
ARM9TDMI 207
ARM9E 207
ARM10 207
XScale 208
ARM11 208
Cortex 208

APPENDIX D: NEON INTRINSICS AND INSTRUCTIONS 215

Data Types 215
Lane Types 216
Assembly Instructions 216
Intrinsic Naming Conventions 220

APPENDIX E: ASSEMBLY INSTRUCTIONS 221

ARM Instructions 221
Thumb Instructions on Cortex-M Cores 234

INDEX *247*

INTRODUCTION

IN THE WORLD OF EMBEDDED SYSTEMS, you can't work for long without working on an ARM CPU. ARM CPUs are known for their low electric power consumption, making them ideal for mobile embedded systems. Since 2012, virtually all PDAs and smartphones contain ARM CPUs, and ARMs account for 75 percent of all 32-bit embedded systems and 90 percent of embedded RISC systems. In 2005, 98 percent of more than one billion mobile phones sold used at least one ARM processor. You can find ARM processors in mobile phones, tablets, MP3 players, handheld games consoles, calculators, and even computer peripherals such as Bluetooth chips and hard disk drives.

With more than 1 billion ARM processors shipped every 2 months, it is surprising to know that ARM does not actually make processors, but rather designs the core, and ARM partners use those designs to make their own processors, adding external devices and peripherals or modifying the core for speed or power consumption benefits. By working closely with manufacturers, ARM has created a huge ecosystem. The result is an amazing range of processors, used for all types of devices in all classes of devices, and all running using a common architecture, enabling developers to switch easily from one processor to another.

ARM processors are by no means tiny processors with limited performance; they range from micro-controller devices used in the smallest of systems all the way to 64-bit processors used in servers.

This book introduces you to embedded ARM systems, how to get them up and running, how to develop for this platform, and some of the devices available in this huge ecosystem.

WHO THIS BOOK IS FOR

This book is primarily for developers who want to start in the embedded field. A basic understanding of C is required for most examples, but no assembly knowledge is required.

This book is also for developers who want better knowledge of the internals of a processor and to understand what goes on deep inside the core.

WHAT THIS BOOK COVERS

This book covers the advances in technology for ARM processors and focuses on the more recent ARMv7 architecture, for Cortex-A, Cortex-R, and Cortex-M devices. If you use the Cortex range of processors, you will feel at home, but if you use ARM Classic cores, you can also find information and a listing of the differences between architectures and platforms.

HOW THIS BOOK IS STRUCTURED

This book is designed to give as much information as possible to someone who does not have working experience with ARM processors. To understand ARM's philosophy, it is necessary to understand where ARM came from and how the ARM processor was born. This book then covers all aspects of an embedded project: understanding the processor and the extensions, understanding assembler, creating your first program using a more familiar C, and then continuing to debugging and optimization.

Chapter 1, "The History of ARM," gives an outline of the fascinating history of ARM; where it came from and why it is where it is today.

Chapter 2, "ARM Embedded Systems," gives an explanation on what an embedded system is and a presentation of the strong points of an ARM system.

Chapter 3, "ARM Architecture," lists the different elements that compose an ARM processor and how to use them effectively.

Chapter 4, "ARM Assembly Language," gives an introduction to ARM assembly and explains why understanding assembly is so important.

Chapter 5, "First Steps," presents some simulators and real-world cards to write programs, both to get an ARM processor started and to use as a basis for more complex programs. This chapter also presents some real-world scenario projects.

Chapter 6, "Thumb Instruction Set," presents the Thumb mode and also the Thumb-2 extension. Cortex-M processors use only Thumb mode, but Thumb can also be used on every modern processor where code density is important.

Chapter 7, "Assembly Instructions," presents a list of assembly instructions in ARM's Unified Assembly Language, and explains their use with easy-to-follow programs.

Chapter 8, "NEON," presents NEON, ARM's advanced Single Instruction Multiple Data processor and shows how you can use it to accelerate mathematically intensive routines.

Chapter 9, "Debugging," describes debugging, what is required to debug a program, and what you can achieve. It uses several real-world examples.

Chapter 10, "Writing Optimized C," describes the final part of any application—optimization. With some simple examples, you will learn how to write optimized code, and understand what happens deep inside the processor to implement further optimization.

Appendix A, "Terminology," explains some of the words and terms you will encounter when working on embedded systems, and more specifically, ARM embedded systems.

Appendix B, "ARM Architecture Versions," lists the different ARM Architectures that exist, and explain what each version brought in terms of technological advancement, but also which processor belongs to which architecture.

Appendix C, "ARM Core Versions," looks closer at the ARM cores, and presents the changes in each processor. Discussing briefly each processor from ARM6 onwards, it goes into more detail for modern Cortex-class processors.

Appendix D, "Neon Intrinsics and Instructions," lists the different instructions available for ARM's NEON engine, and also presents the intrinsics used to perform NEON calculation in an optimized way, using C.

Appendix E, "Assembly Instructions," lists the different assembly instructions used in UAL, with a description of each, as well as a list of Thumb instructions used on different Cortex-M class processors.

WHAT YOU NEED TO USE THIS BOOK

Most people imagine an embedded system surrounded with costly electronics and expensive software licenses, but the truth is that you can start embedded development with little investment. To start, you need a development computer. Examples are given for Linux, but you can also use Windows and MacOS. Royalty-free compilers are proposed, and you can use a free ARM simulator for your first programs, but later, a small ARM system is advisable: either an evaluation board from a manufacturer (two are presented), or you can use an inexpensive small-factor computer, such as a Raspberry Pi or an Arduino Due.

To run the samples in the book, you need the following:

- Linux development computer
- Mentor Graphics compiler suite
- Atmel SAM D20 Xplained Pro evaluation board
- Silicon Lab's STK3200 and STK3800 evaluation boards
- Raspberry Pi

The source code for the samples is available for download from the Wrox website at:

```
www.wiley.com/go/profembeddedarmdev
```

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

> **NOTE** *Notes indicate notes, tips, hints, tricks, and asides to the current discussion.*

As for styles in the text:

- We *highlight* new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show filenames, URLs, and code within the text like so: `persistence.properties`.
- We present code in this way:

```
We use a monofont type with no highlighting for most code examples.
```

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All the source code used in this book is available for download at `www.wrox.com`. Specifically for this book, the code download is on the Download Code tab at:

`www.wrox.com/go/profembeddedarmdev`

You can also search for the book at `www.wrox.com` by ISBN (the ISBN for this book is 978-1-118-78894-3) to find the code. And a complete list of code downloads for all current Wrox books is available at `www.wrox.com/dynamic/books/download.aspx`.

At the beginning of Chapter 5, you will find a list of the major code files for the chapter. Throughout the chapter, you will also find references to the names of code files available for download.

Most of the code on `www.wrox.com` is compressed in a .ZIP, .RAR archive, or similar archive format appropriate to the platform.

After you download the code, decompress it with your favorite compression tool. Alternatively, you can go to the main Wrox code download page at `www.wrox.com/dynamic/books/download.aspx` to see the code available for this book and all other Wrox books.

ERRATA

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time, you can help us provide higher quality information.

To find the errata page for this book, go to:

`www.wrox.com/go/profembeddedarmdev`

Then click the Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors.

If you don't spot "your" error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

P2P.WROX.COM

For author and peer discussion, join the P2P forums at http://p2p.wrox.com. The forums are a web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At http://p2p.wrox.com, you will find a number of different forums that will help you, not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to http://p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join, as well as any optional information you want to provide, and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

NOTE *You can read messages in the forums without joining P2P, but to post your own messages, you must join.*

After you join, you can post new messages and respond to messages other users post. You can read messages at any time on the web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works, as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

PART I
ARM Systems and Development

➤ **CHAPTER 1:** The History of ARM

➤ **CHAPTER 2:** ARM Embedded Systems

➤ **CHAPTER 3:** ARM Architecture

➤ **CHAPTER 4:** ARM Assembly Language

➤ **CHAPTER 5:** First Steps

➤ **CHAPTER 6:** Thumb Instruction Set

➤ **CHAPTER 7:** Assembly Instructions

➤ **CHAPTER 8:** NEON

➤ **CHAPTER 9:** Debugging

➤ **CHAPTER 10:** Writing Optimized C