Professional Assembly Language

Richard Blum

Wiley Publishing, Inc.

Professional Assembly Language

Professional Assembly Language

Richard Blum

Wiley Publishing, Inc.

Professional Assembly Language

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published simultaneously in Canada

ISBN: 0-7645-7901-0

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1MA/SW/QR/QV/IN

Blum, Richard. 1962-
Professional assembly language / Richard Blum.
p. cm.
Includes index.
ISBN 0-7645-7901-0 (paper/website)
1. Assembly language (Computer program language) 1. Title.
QA76.73.A8B58 2005
005.13'6—dc22

2004029116

About the Author

Richard Blum has worked for a large U.S. government organization for more than 15 years. During that time, he has had the opportunity to program utilities in various programming languages: C, C++, Java, and Microsoft VB.NET and C#. With this experience, Rich has often found the benefit of reviewing assembly language code generated by compilers and utilizing assembly language routines to speed up higher-level language programs.

Rich has a bachelor of science degree in electrical engineering from Purdue University, where he worked on many assembly language projects. (Of course, this was back in the eight-bit processor days.) He also has a master of science degree in management from Purdue University, specializing in Management Information Systems.

When Rich is not being a computer nerd, he is either playing electric bass for the church worship band or spending time with his wife, Barbara, and two daughters, Katie Jane and Jessica.

Credits

Executive Editor
Chris Webb

Development Editor
Adaobi Obi Tulton

Production Editor
William A. Barton

Technical Editor
Paul Carter

Copy Editor
Luann Rouff

Editorial Manager
Kathryn Malm Bourgoine

Vice President & Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Coordinator
Erin Smith

Graphics and Production Specialists
Jonelle Burns
Amanda Carter
Carrie A. Foster
Lauren Goddard
Denny Hager
Joyce Haughey

Quality Control Technicians
David Faust
Susan Moritz
Carl William Pierce

Media Development Specialist
Angie Denny

Proofreading
TECHBOOKS Production Services

Indexing
Richard T. Evans

This book is dedicated to my wife, Barbara, and my daughters, Katie Jane and Jessica. “Trust in the Lord with all your heart and lean not on your own understanding; in all ways acknowledge him, and he will make your paths straight.” Pr 3:5-6 (NIV)

Acknowledgments

First, all honor, glory, and praise go to God, who through His Son makes all things possible and gives us the gift of eternal life.

Many thanks go to the great team of people at John Wiley & Sons Publishing. Thanks to Chris Webb, the acquisitions editor, for offering me the opportunity to write this book. I am forever indebted to Adaobi Obi Tulton, the development editor, for her work in making this book presentable and her overall guidance through the book writing process. Also, many thanks go to Paul Carter, the technical editor of the book. Paul's comments throughout the book were invaluable in presenting the topic in the best way and for pointing out my goofs and blunders. I would also like to thank Carole McClendon at Waterside Productions, Inc., for arranging this opportunity for me, and for helping out in my writing career.

Finally, I would like to thank my parents, Mike and Joyce Blum, for their dedication and support while raising me, and to my wife, Barbara, and daughters, Katie Jane and Jessica, for their love, patience, and understanding, especially while I was writing this book.

Contents

Introduction

Assembly language is one of the most misunderstood programming languages in use. When the term assembly language is used, it often invokes the idea of low-level bit shuffling and poring over thousand-page instruction manuals looking for the proper instruction format. With the proliferation of fancy high-level language development tools, it is not uncommon to see the phrase "assembly language programming is dead" pop up among various programming newsgroups.

However, assembly language programming is far from dead. Every high-level language program must be compiled into assembly language before it can be linked into an executable program. For the high-level language programmer, understanding how the compiler generates the assembly language code can be a great benefit, both for directly writing routines in assembly language and for understanding how the high-level language routines are converted to assembly language by the compiler.

Who This Book Is For

The primary purpose of this book is to teach high-level language programmers how their programs are converted to assembly language, and how the generated assembly language code can be tweaked. That said, the main audience for this book is programmers already familiar with a high-level language, such as C, C++, or even Java. This book does not spend much time teaching basic programming principles. It assumes that you are already familiar with the basics of computer programming, and are interested in learning assembly language to understand what is happening underneath the hood.

However, if you are new to programming and are looking at assembly language programming as a place to start, this book does not totally ignore you. It is possible to follow along in the chapters from the start to the finish and obtain a basic knowledge of how assembly language programming (and programming in general) works. Each of the topics presented includes example code that demonstrates how the assembly language instructions work. If you are completely new to programming, I recommend that you also obtain a good introductory text to programming to round out your education on the topic.

What This Book Covers

The main purpose of this book is to familiarize C and C++ programmers with assembly language, show how compilers create assembly language routines from C and C++ programs, and show how the generated assembly language routines can be spruced up to increase the performance of an application.

All high-level language programs (such as C and C++) are converted to assembly language by the compiler before being linked into an executable program. The compiler uses specific rules defined by the creator of the compiler to determine exactly how the high-level language statements are converted. Many programmers just write their high-level language programs and assume the compiler is creating the proper executable code to implement the program.

However, this is not always the case. When the compiler converts the high-level language code statements into assembly language code, quirks and oddities often pop up. In addition, the compiler is often written to follow the conversion rules so specifically that it does not recognize time-saving shortcuts that can be made in the final assembly language code, or it is unable to compensate for poorly written high-level routines. This is where knowledge of assembly language code can come in handy.

This book shows that by examining the assembly language code generated by the compiler before linking it into an executable program, you can often find places where the code can be modified to increase performance or provide additional functionality. The book also helps you understand how your high-level language routines are affected by the compiler's conversion process.

How This Book Is Structured

The book is divided into three sections. The first section covers the basics of the assembly language programming environment. Because assembly language programming differs among processors and assemblers, a common platform had to be chosen. This book uses the popular Linux operating system, running on the Intel family of processors. The Linux environment provides a wealth of program developer tools, such as an optimizing compiler, an assembler, a linker, and a debugger, all at little or no charge. This wealth of development tools in the Linux environment makes it the perfect setting for dissecting C programs into assembly language code.

The chapters in the first section are as follows:

Chapter 1, "What Is Assembly Language?" starts the section off by ensuring that you understand exactly what assembly language is and how it fits into the programming model. It debunks some of the myths of assembly language, and provides a basis for understanding how to use assembly language with high-level languages.

Chapter 2, "The IA-32 Platform," provides a brief introduction to the Intel Pentium family of processors. When working with assembly language, it is important that you understand the underlying processor and how it handles programs. While this chapter is not intended to be an in-depth analysis of the operation of the IA-32 platform, it does present the hardware and operations involved with programming for that platform.

Chapter 3, "The Tools of the Trade," presents the Linux open-source development tools that are used throughout the book. The GNU compiler, assembler, linker, and debugger are used in the book for compiling, assembling, linking, and debugging the programs.

Chapter 4, "A Sample Assembly Language Program," demonstrates how to use the GNU tools on a Linux system to create, assemble, link, and debug a simple assembly language program. It also shows how to use C library functions within assembly language programs on Linux systems to add extra features to your assembly language applications.

The second section of the book dives into the basics of assembly language programming. Before you can start to analyze the assembly language code generated by the compiler, you must understand the assembly language instructions. The chapters in this section are as follows:

Chapter 5, "Moving Data," shows how data elements are moved in assembly language programs. The concepts of registers, memory locations, and the stack are presented, and examples are shown for moving data between them.

Chapter 6, "Controlling Execution Flow," describes the branching instructions used in assembly language programs. Possibly one of the most important features of programs, the ability to recognize branches and optimize branches is crucial to increasing the performance of an application.

Chapter 7, "Using Numbers," discusses how different number data types are used in assembly language. Being able to properly handle integers and floating-point values is important within the assembly language program.

Chapter 8, "Basic Math Functions," shows how assembly language instructions are used to perform the basic math functions such as addition, subtraction, multiplication, and division. While these are generally straightforward functions, subtle tricks can often be used to increase performance in this area.

Chapter 9, "Advanced Math Functions," discusses the IA-32 Floating Point Unit (FPU), and how it is used to handle complex floating-point arithmetic. Floating-point arithmetic is often a crucial element to data processing programs, and knowing how it works greatly benefits high-level language programmers.

Chapter 10, "Working with Strings," presents the various assembly language string-handling instructions. Character data is another important facet of high-level language programming. Understanding how the assembly language level handles strings can provide insights when working with strings in high-level languages.

Chapter 11, "Using Functions," begins the journey into the depths of assembly language programming. Creating assembly language functions to perform routines is at the core of assembly language optimization. It is good to know the basics of assembly language functions, as they are often used by the compiler when generating the assembly language code from high-level language code.

Chapter 12, "Using Linux System Calls," completes this section by showing how many high-level functions can be performed in assembly language using already created functions. The Linux system provides many high-level functions, such as writing to the display. Often, you can utilize these functions within your assembly language program.

The last section of the book presents more advanced assembly language topics. Because the main topic of this book is how to incorporate assembly language routines in your C or C++ code, the first few chapters show just how this is done. The remaining chapters present some more advanced topics to round out your education on assembly language programming. The chapters in this section include the following:

Chapter 13, "Using Inline Assembly," shows how to incorporate assembly language routines directly in your C or C++ language programs. Inline assembly language is often used for "hard-coding" quick routines in the C program to ensure that the compiler generates the appropriate assembly language code for the routine.

Chapter 14, "Calling Assembly Libraries," demonstrates how assembly language functions can be combined into libraries that can be used in multiple applications (both assembly language and high-level language). It is a great time-saving feature to be able to combine frequently used functions into a single library that can be called by C or C++ programs.

Chapter 15, "Optimizing Routines," discusses the heart of this book: modifying compiler-generated assembly language code to your taste. This chapter shows exactly how different types of C routines (such as if-then statements and for-next loops) are produced in assembly language code. Once you understand what the assembly language code is doing, you can add your own touches to it to customize the code for your specific environment.

Chapter 16, "Using Files," covers one of the most overlooked functions of assembly language programming. Almost every application requires some type of file access on the system. Assembly language programs are no different. This chapter shows how to use the Linux file-handling system calls to read, write, and modify data in files on the system.

Chapter 17, "Using Advanced IA-32 Features," completes the book with a look at the advanced Intel Single Instruction Multiple Data (SIMD) technology. This technology provides a platform for programmers to perform multiple arithmetic operations in a single instruction. This technology has become crucial in the world of audio and video data processing.

What You Need to Use This Book

All of the examples in this book are coded to be assembled and run on the Linux operating system, running on an Intel processor platform. The Open Source GNU compiler (`gcc`), assembler (`gas`), linker (`ld`), and debugger (`gdb`) are used extensively throughout the book to demonstrate the assembly language features. Chapter 4, "A Sample Assembly Language Program," discusses specifically how to use these tools on a Linux platform to create, assemble, link, and debug an assembly language program. If you do not have an installed Linux platform available, Chapter 4 demonstrates how to use a Linux distribution that can be booted directly from CD, without modifying the workstation hard drive. All of the GNU development tools used in this book are available without installing Linux on the workstation.

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.

Tips, hints, tricks, and asides to the current discussion are offset and placed in italics like this.

As for styles in the text:

- We *highlight* important words when we introduce them.
- We show filenames, URLs, and code within the text like so: `persistence.properties`.
- We present code in two different ways:

```
In code examples we highlight new and important code with a gray background.

The gray highlighting is not used for code that's less important in the present
context, or has been shown before.
```

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 of the source code used in this book is available for download at `www.wrox.com`. Once at the site, simply locate the book's title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book's detail page to obtain all the source code for the book.

> *Because many books have similar titles, you may find it easiest to search by ISBN; this book's ISBN is 0-764-57901-0.*

Once you download the code, just decompress it with your favorite compression tool. Alternately, 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, such as a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata, you may save another reader hours of frustration, and at the same time you will be helping us provide even higher quality information.

To find the errata page for this book, go to `www.wrox.com` and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page, you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list, including links to each book's errata, is also available at `www.wrox.com/misc-pages/booklist.shtml`.

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 `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 `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 wish 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.

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

Once 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.