

 WILEY

Jeff Duntemann

# Assembly Language Step by Step

Programming with Linux®

THIRD EDITION





---

# **Assembly Language Step-by-Step**





---

# Assembly Language Step-by-Step

---

Programming with Linux<sup>®</sup>

Third Edition

Jeff Duntemann



WILEY

Wiley Publishing, Inc.

---

## Assembly Language Step-by-Step

Published by  
Wiley Publishing, Inc.  
10475 Crosspoint Boulevard  
Indianapolis, IN 46256  
[www.wiley.com](http://www.wiley.com)

Copyright © 2009 by Jeff Duntemann

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-49702-9

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

**Library of Congress Control Number:** 2009933745

**Trademarks:** Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Linux is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book.

---

*To the eternal memory of  
Kathleen M. Duntemann, Godmother  
1920–1999  
who gave me books when all I could do was put teeth marks on them.  
There are no words for how much I owe you!*





---



# About the Author

**Jeff Duntemann** is a writer, editor, lecturer, and publishing industry analyst. In his thirty years in the technology industry he has been a computer programmer and systems analyst for Xerox Corporation, a technical journal editor for Ziff-Davis Publications, and Editorial Director for Coriolis Group Books and later Paraglyph Press. He is currently a technical publishing consultant and also owns Copperwood Press, a POD imprint hosted on lulu.com. Jeff lives with his wife Carol in Colorado Springs, Colorado.



---



# Credits

**Executive Editor**

Carol Long

**Project Editor**

Brian Herrmann

**Production Editor**

Rebecca Anderson

**Copy Editor**

Luann Rouff

**Editorial Director**

Robyn B. Siesky

**Editorial Manager**

Mary Beth Wakefield

**Production Manager**

Tim Tate

**Vice President and Executive****Group Publisher**

Richard Swadley

**Vice President and Executive  
Publisher**

Barry Pruett

**Associate Publisher**

Jim Minatel

**Project Coordinator, Cover**

Lynsey Stanford

**Proofreader**

Dr. Nate Pritts, Word One

**Indexer**

J&J Indexing

**Cover Image**

© Jupiter Images/Corbis/  
Lawrence Manning



---



# Acknowledgments

First of all, thanks are due to Carol Long and Brian Herrmann at Wiley, for allowing this book another shot, and then making sure it happened, on a much more aggressive schedule than last time.

As for all three previous editions, I owe Michael Abrash a debt of gratitude for constant sane advice on many things, especially the arcane differences between modern Intel microarchitectures.

Although they might not realize it, Randy Hyde, Frank Kotler, Beth, and all the rest of the gang on alt.lang.asm were very helpful in several ways, not least of which was hearing and answering requests from assembly language newcomers, thus helping me decide what must be covered in a book like this and what need not.

Finally, and as always, a toast to Carol for the support and sacramental friendship that has enlivened me now for 40 years, and enabled me to take on projects like this and see them through to the end.



---



# Contents at a Glance

|  |              |
|--|--------------|
| <b>Introduction: “Why Would You Want to Do <i>That?</i>”</b> | <b>xxvii</b> |
| <b>Chapter 1 Another Pleasant Valley Saturday</b>            | <b>1</b>     |
| <b>Chapter 2 Alien Bases</b>                                 | <b>15</b>    |
| <b>Chapter 3 Lifting the Hood</b>                            | <b>45</b>    |
| <b>Chapter 4 Location, Location, Location</b>                | <b>77</b>    |
| <b>Chapter 5 The Right to Assemble</b>                       | <b>109</b>   |
| <b>Chapter 6 A Place to Stand, with Access to Tools</b>      | <b>155</b>   |
| <b>Chapter 7 Following Your Instructions</b>                 | <b>201</b>   |
| <b>Chapter 8 Our Object All Sublime</b>                      | <b>237</b>   |
| <b>Chapter 9 Bits, Flags, Branches, and Tables</b>           | <b>279</b>   |
| <b>Chapter 10 Dividing and Conquering</b>                    | <b>327</b>   |
| <b>Chapter 11 Strings and Things</b>                         | <b>393</b>   |
| <b>Chapter 12 Heading Out to C</b>                           | <b>439</b>   |
| <b>Conclusion: Not the End, But Only the Beginning</b>       | <b>503</b>   |
| <b>Appendix A Partial x86 Instruction Set Reference</b>      | <b>507</b>   |
| <b>Appendix B Character Set Charts</b>                       | <b>583</b>   |
| <b>Index</b>   | <b>587</b>   |





---



# Contents

|  |              |
|--|--------------|
| <b>Introduction: “Why Would You Want to Do <i>That?</i>”</b> | <b>xxvii</b> |
| <b>Chapter 1 Another Pleasant Valley Saturday</b>            | <b>1</b>     |
| It’s All in the Plan   | 1            |
| Steps and Tests  | 2            |
| More Than Two Ways?  | 3            |
| Computers Think Like Us                                      | 4            |
| Had This Been the Real Thing . . .                           | 4            |
| Do Not Pass Go   | 5            |
| The Game of Big Bux  | 6            |
| Playing Big Bux  | 8            |
| Assembly Language Programming As a Board Game                | 9            |
| Code and Data  | 10           |
| Addresses  | 11           |
| Metaphor Check!  | 12           |
| <b>Chapter 2 Alien Bases</b>                                 | <b>15</b>    |
| The Return of the New Math Monster                           | 15           |
| Counting in Martian  | 16           |
| Dissecting a Martian Number                                  | 18           |
| The Essence of a Number Base                                 | 20           |
| Octal: How the Grinch Stole Eight and Nine                   | 20           |
| Who Stole Eight and Nine?                                    | 21           |
| Hexadecimal: Solving the Digit Shortage                      | 24           |
| From Hex to Decimal and from Decimal to Hex                  | 28           |
| From Hex to Decimal  | 28           |
| From Decimal to Hex  | 29           |
| Practice. Practice! PRACTICE!                                | 31           |

|  |           |
|--|-----------|
| Arithmetic in Hex                                | 32        |
| Columns and Carries                              | 35        |
| Subtraction and Borrows                          | 35        |
| Borrows across Multiple Columns                  | 37        |
| What's the Point?                                | 38        |
| Binary   | 38        |
| Values in Binary                                 | 40        |
| Why Binary?                                      | 42        |
| Hexadecimal As Shorthand for Binary              | 43        |
| Prepare to Compute                               | 44        |
| <b>Chapter 3 Lifting the Hood</b>                | <b>45</b> |
| RAXie, We Hardly Knew Ye . . .                   | 45        |
| Gus to the Rescue                                | 46        |
| Switches, Transistors, and Memory                | 47        |
| One If by Land . . .                             | 48        |
| Transistor Switches                              | 48        |
| The Incredible Shrinking Bit                     | 50        |
| Random Access                                    | 52        |
| Memory Access Time                               | 53        |
| Bytes, Words, Double Words, and Quad Words       | 54        |
| Pretty Chips All in a Row                        | 55        |
| The Shop Foreman and the Assembly Line           | 57        |
| Talking to Memory                                | 58        |
| Riding the Data Bus                              | 59        |
| The Foreman's Pockets                            | 60        |
| The Assembly Line                                | 61        |
| The Box That Follows a Plan                      | 61        |
| Fetch and Execute                                | 63        |
| The Foreman's Innards                            | 64        |
| Changing Course                                  | 65        |
| What vs. How: Architecture and Microarchitecture | 66        |
| Evolving Architectures                           | 67        |
| The Secret Machinery in the Basement             | 68        |
| Enter the Plant Manager                          | 70        |
| Operating Systems: The Corner Office             | 70        |
| BIOS: Software, Just Not as Soft                 | 71        |
| Multitasking Magic                               | 71        |
| Promotion to Kernel                              | 73        |
| The Core Explosion                               | 73        |
| The Plan   | 74        |

|                  |   |            |
|------------------|---|------------|
| <b>Chapter 4</b> | <b>Location, Location, Location</b>                                   | <b>77</b>  |
|                  | The Joy of Memory Models  | 77         |
|                  | 16 Bits'll Buy You 64K  | 79         |
|                  | The Nature of a Megabyte  | 82         |
|                  | Backward Compatibility and Virtual 86 Mode                            | 83         |
|                  | 16-Bit Blinders   | 83         |
|                  | The Nature of Segments  | 85         |
|                  | A Horizon, Not a Place  | 88         |
|                  | Making 20-Bit Addresses out of 16-Bit Registers                       | 88         |
|                  | 16-Bit and 32-Bit Registers   | 90         |
|                  | General-Purpose Registers   | 91         |
|                  | Register Halves   | 93         |
|                  | The Instruction Pointer   | 95         |
|                  | The Flags Register  | 96         |
|                  | The Three Major Assembly Programming Models                           | 96         |
|                  | Real Mode Flat Model  | 97         |
|                  | Real Mode Segmented Model   | 99         |
|                  | Protected Mode Flat Model   | 101        |
|                  | What Protected Mode Won't Let Us Do Anymore                           | 104        |
|                  | Memory-Mapped Video   | 104        |
|                  | Direct Access to Port Hardware  | 105        |
|                  | Direct Calls into the BIOS  | 106        |
|                  | Looking Ahead: 64-Bit "Long Mode"                                     | 106        |
|                  | 64-Bit Memory: What May Be Possible Someday vs.<br>What We Can Do Now | 107        |
| <b>Chapter 5</b> | <b>The Right to Assemble</b>  | <b>109</b> |
|                  | Files and What's Inside Them  | 110        |
|                  | Binary Files vs. Text Files   | 111        |
|                  | Looking at File Internals with the Bless Editor                       | 112        |
|                  | Interpreting Raw Data   | 116        |
|                  | "Endianness"  | 117        |
|                  | Text In, Code Out   | 121        |
|                  | Assembly Language   | 121        |
|                  | Comments  | 124        |
|                  | Beware "Write-Only" Source Code!                                      | 124        |
|                  | Object Code and Linkers   | 125        |
|                  | Relocatability  | 128        |
|                  | The Assembly Language Development Process                             | 128        |
|                  | The Discipline of Working Directories                                 | 129        |
|                  | Editing the Source Code File  | 131        |

|   |            |
|---|------------|
| Assembling the Source Code File                         | 131        |
| Assembler Errors  | 132        |
| Back to the Editor                                      | 133        |
| Assembler Warnings                                      | 134        |
| Linking the Object Code File                            | 135        |
| Linker Errors   | 136        |
| Testing the .EXE File                                   | 136        |
| Errors versus Bugs                                      | 137        |
| Are We There Yet?                                       | 138        |
| Debuggers and Debugging                                 | 138        |
| Taking a Trip Down Assembly Lane                        | 139        |
| Installing the Software                                 | 139        |
| Step 1: Edit the Program in an Editor                   | 142        |
| Step 2: Assemble the Program with NASM                  | 143        |
| Step 3: Link the Program with LD                        | 146        |
| Step 4: Test the Executable File                        | 147        |
| Step 5: Watch It Run in the Debugger                    | 147        |
| Ready to Get Serious?                                   | 153        |
| <b>Chapter 6 A Place to Stand, with Access to Tools</b> | <b>155</b> |
| The Kate Editor   | 157        |
| Installing Kate   | 157        |
| Launching Kate  | 158        |
| Configuration   | 160        |
| Kate Sessions   | 162        |
| Creating a New Session                                  | 162        |
| Opening an Existing Session                             | 163        |
| Deleting or Renaming Sessions                           | 163        |
| Kate's File Management                                  | 164        |
| Filesystem Browser Navigation                           | 165        |
| Adding a File to the Current Session                    | 165        |
| Dropping a File from the Current Session                | 166        |
| Switching Between Session Files in the Editor           | 166        |
| Creating a Brand-New File                               | 166        |
| Creating a Brand-New Folder on Disk                     | 166        |
| Deleting a File from Disk (Move File to Trash)          | 166        |
| Reloading a File from Disk                              | 167        |
| Saving All Unsaved Changes in Session Files             | 167        |
| Printing the File in the Editor Window                  | 167        |
| Exporting a File As HTML                                | 167        |
| Adding Items to the Toolbar                             | 167        |
| Kate's Editing Controls                                 | 168        |
| Cursor Movement   | 169        |
| Bookmarks   | 169        |
| Selecting Text  | 170        |

|  |            |
|--|------------|
| Searching the Text                           | 171        |
| Using Search and Replace                     | 172        |
| Using Kate While Programming                 | 172        |
| Creating and Using Project Directories       | 173        |
| Focus!                                       | 175        |
| Linux and Terminals                          | 176        |
| The Linux Console                            | 176        |
| Character Encoding in Konsole                | 177        |
| The Three Standard Unix Files                | 178        |
| I/O Redirection                              | 180        |
| Simple Text Filters                          | 182        |
| Terminal Control with Escape Sequences       | 183        |
| So Why Not GUI Apps?                         | 185        |
| Using Linux Make                             | 186        |
| Dependencies                                 | 187        |
| When a File Is Up to Date                    | 189        |
| Chains of Dependencies                       | 189        |
| Invoking Make from Inside Kate               | 191        |
| Using Touch to Force a Build                 | 193        |
| The Insight Debugger                         | 194        |
| Running Insight                              | 195        |
| Insight's Many Windows                       | 195        |
| A Quick Insight Run-Through                  | 197        |
| Pick Up Your Tools . . .                     | 200        |
| <b>Chapter 7 Following Your Instructions</b> | <b>201</b> |
| Build Yourself a Sandbox                     | 201        |
| A Minimal NASM Program                       | 202        |
| Instructions and Their Operands              | 204        |
| Source and Destination Operands              | 204        |
| Immediate Data                               | 205        |
| Register Data                                | 207        |
| Memory Data                                  | 209        |
| Confusing Data and Its Address               | 210        |
| The Size of Memory Data                      | 211        |
| The Bad Old Days                             | 211        |
| Rally Round the Flags, Boys!                 | 212        |
| Flag Etiquette                               | 215        |
| Adding and Subtracting One with INC and DEC  | 215        |
| Watching Flags from Insight                  | 216        |
| How Flags Change Program Execution           | 218        |
| Signed and Unsigned Values                   | 221        |
| Two's Complement and NEG                     | 221        |
| Sign Extension and MOVSX                     | 224        |

|  |            |
|--|------------|
| Implicit Operands and MUL                            | 225        |
| MUL and the Carry Flag                               | 227        |
| Unsigned Division with DIV                           | 228        |
| The x86 Slowpokes                                    | 229        |
| Reading and Using an Assembly Language Reference     | 230        |
| Memory Joggers for Complex Memories                  | 230        |
| An Assembly Language Reference for Beginners         | 231        |
| Flags  | 232        |
| NEG: Negate (Two's Complement; i.e., Multiply by -1) | 233        |
| Flags affected                                       | 233        |
| Legal forms  | 233        |
| Examples   | 233        |
| Notes  | 233        |
| Legal Forms  | 234        |
| Operand Symbols                                      | 234        |
| Examples   | 235        |
| Notes  | 235        |
| What's Not Here . . .                                | 235        |
| <b>Chapter 8 Our Object All Sublime</b>              | <b>237</b> |
| The Bones of an Assembly Language Program            | 237        |
| The Initial Comment Block                            | 239        |
| The .data Section                                    | 240        |
| The .bss Section                                     | 240        |
| The .text Section                                    | 241        |
| Labels   | 241        |
| Variables for Initialized Data                       | 242        |
| String Variables                                     | 242        |
| Deriving String Length with EQU and \$               | 244        |
| Last In, First Out via the Stack                     | 246        |
| Five Hundred Plates per Hour                         | 246        |
| Stacking Things Upside Down                          | 248        |
| Push-y Instructions                                  | 249        |
| POP Goes the Opcode                                  | 251        |
| Storage for the Short Term                           | 253        |
| Using Linux Kernel Services Through INT80            | 254        |
| An Interrupt That Doesn't Interrupt Anything         | 254        |
| Getting Home Again                                   | 259        |
| Exiting a Program via INT 80h                        | 260        |
| Software Interrupts versus Hardware Interrupts       | 261        |
| INT 80h and the Portability Fetish                   | 262        |
| Designing a Non-Trivial Program                      | 264        |
| Defining the Problem                                 | 264        |
| Starting with Pseudo-code                            | 265        |

|  |            |
|--|------------|
| Successive Refinement                        | 266        |
| Those Inevitable “Whoops!” Moments           | 270        |
| Scanning a Buffer                            | 271        |
| “Off By One” Errors                          | 273        |
| Going Further                                | 277        |
| <b>Chapter 9</b>                             | <b>279</b> |
| <b>Bits, Flags, Branches, and Tables</b>     | <b>279</b> |
| Bits Is Bits (and Bytes Is Bits)             | 279        |
| Bit Numbering                                | 280        |
| “It’s the Logical Thing to Do, Jim. . .”     | 280        |
| The AND Instruction                          | 281        |
| Masking Out Bits                             | 282        |
| The OR Instruction                           | 283        |
| The XOR Instruction                          | 284        |
| The NOT Instruction                          | 285        |
| Segment Registers Don’t Respond to Logic!    | 285        |
| Shifting Bits                                | 286        |
| Shift By What?                               | 286        |
| How Bit Shifting Works                       | 287        |
| Bumping Bits into the Carry Flag             | 287        |
| The Rotate Instructions                      | 288        |
| Setting a Known Value into the Carry Flag    | 289        |
| Bit-Bashing in Action                        | 289        |
| Splitting a Byte into Two Nybbles            | 292        |
| Shifting the High Nybble into the Low Nybble | 293        |
| Using a Lookup Table                         | 293        |
| Multiplying by Shifting and Adding           | 295        |
| Flags, Tests, and Branches                   | 298        |
| Unconditional Jumps                          | 298        |
| Conditional Jumps                            | 299        |
| Jumping on the Absence of a Condition        | 300        |
| Flags  | 301        |
| Comparisons with CMP                         | 301        |
| A Jungle of Jump Instructions                | 302        |
| “Greater Than” Versus “Above”                | 303        |
| Looking for 1-Bits with TEST                 | 304        |
| Looking for 0 Bits with BT                   | 306        |
| Protected Mode Memory Addressing in Detail   | 307        |
| Effective Address Calculations               | 308        |
| Displacements                                | 309        |
| Base + Displacement Addressing               | 310        |
| Base + Index Addressing                      | 310        |
| Index × Scale + Displacement Addressing      | 312        |
| Other Addressing Schemes                     | 313        |

|   |            |
|---|------------|
| LEA: The Top-Secret Math Machine                          | 315        |
| The Burden of 16-Bit Registers                            | 317        |
| Character Table Translation                               | 318        |
| Translation Tables  | 318        |
| Translating with MOV or XLAT                              | 320        |
| Tables Instead of Calculations                            | 325        |
| <b>Chapter 10 Dividing and Conquering</b>                 | <b>327</b> |
| Boxes within Boxes  | 328        |
| Procedures As Boxes for Code                              | 329        |
| Calling and Returning                                     | 336        |
| Calls within Calls  | 338        |
| The Dangers of Accidental Recursion                       | 340        |
| A Flag Etiquette Bug to Beware Of                         | 341        |
| Procedures and the Data They Need                         | 342        |
| Saving the Caller's Registers                             | 343        |
| Local Data  | 346        |
| More Table Tricks   | 347        |
| Placing Constant Data in Procedure Definitions            | 349        |
| Local Labels and the Lengths of Jumps                     | 350        |
| "Forcing" Local Label Access                              | 353        |
| Short, Near, and Far Jumps                                | 354        |
| Building External Procedure Libraries                     | 355        |
| Global and External Declarations                          | 356        |
| The Mechanics of Globals and Externals                    | 357        |
| Linking Libraries into Your Programs                      | 365        |
| The Dangers of Too Many Procedures and Too Many Libraries | 366        |
| The Art of Crafting Procedures                            | 367        |
| Maintainability and Reuse                                 | 367        |
| Deciding What Should Be a Procedure                       | 368        |
| Use Comment Headers!                                      | 370        |
| Simple Cursor Control in the Linux Console                | 371        |
| Console Control Cautions                                  | 377        |
| Creating and Using Macros                                 | 378        |
| The Mechanics of Macro Definition                         | 379        |
| Defining Macros with Parameters                           | 385        |
| The Mechanics of Invoking Macros                          | 386        |
| Local Labels Within Macros                                | 387        |
| Macro Libraries As Include Files                          | 388        |
| Macros versus Procedures: Pros and Cons                   | 389        |



- **Episodes of Life book**
- read First to Kill
- Holy Superheroes!: Exploring the Sacred in Comics, Graphic Novels, and Film (Revised and Expanded Edition) pdf, azw (kindle)
- Judith: A Novel book
  
- <http://korplast.gr/lib/Super-Natural-Every-Day--Well-Loved-Recipes-from-My-Natural-Foods-Kitchen.pdf>
- <http://cavalldecartro.highlandagency.es/library/First-to-Kill.pdf>
- <http://schroff.de/books/La-tranquillit---de-l---me.pdf>
- <http://weddingcellist.com/lib/Judith--A-Novel.pdf>