



Defect



Detect

Windows Debugging Disassembling Reversing

Second Edition

Practical Foundations
Training Course

Dmitry Vostokov
Software Diagnostics Services

Windows Debugging, Disassembling, Reversing

Practical Foundations: Training Course

Second Edition

Dmitry Vostokov
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

Copyright © 2022 by Dmitry Vostokov

Copyright © 2022 by Software Diagnostics Services

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission of the publisher.

OpenTask books are available through booksellers and distributors worldwide. For further information or comments, send requests to press@opentask.com.

Product and company names mentioned in this book may be trademarks of their owners.

A CIP catalog record for this book is available from the British Library.

ISBN-13: 978-1-912636-35-8

Revision 2.5 (April 2022)

Summary of Contents

Contents.....	5
Preface to the Second Edition.....	15
Preface to the First Edition.....	16
Combined Preface from Original Editions.....	17
About the Author	18
Chapter x86.1: Memory, Registers, and Simple Arithmetic	19
Chapter x86.2: Debug and Release Binaries.....	33
Chapter x86.3: Number Representations.....	46
Chapter x86.4: Pointers	53
Chapter x86.5: Bytes, Words, and Double Words	68
Chapter x86.6: Pointers to Memory.....	73
Chapter x86.7: Logical Instructions and EIP	95
Chapter x86.8: Reconstructing a Program with Pointers	102
Chapter x86.9: Memory and Stacks.....	109
Chapter x86.10: Frame Pointer and Local Variables	128
Chapter x86.11: Function Parameters.....	141
Chapter x86.12: More Instructions	153
Chapter x86.13: Function Pointer Parameters.....	164
Chapter x86.14: Summary of Code Disassembly Patterns	170
Appendix x86: Using Docker Environment	174
Chapter x64.1: Memory, Registers, and Simple Arithmetic	176
Chapter x64.2: Debug and Release Binaries.....	191
Chapter x64.3: Number Representations.....	204
Chapter x64.4: Pointers	211
Chapter x64.5: Bytes, Words, and Double Words	229

Chapter x64.6: Pointers to Memory.....	235
Chapter x64.7: Logical Instructions and EIP	258
Chapter x64.8: Reconstructing a Program with Pointers	266
Chapter x64.9: Memory and Stacks.....	275
Chapter x64.10: Local Variables.....	295
Chapter x64.11: Function Parameters.....	305
Chapter x64.12: More Instructions	314
Chapter x64.13: Function Pointer Parameters.....	325
Chapter x64.14: Summary of Code Disassembly Patterns	329
Appendix x64: Using Docker Environment	335

Contents

Contents.....	5
Preface to the Second Edition.....	15
Preface to the First Edition.....	16
Combined Preface from Original Editions.....	17
About the Author	18
Chapter x86.1: Memory, Registers, and Simple Arithmetic	19
Memory and Registers inside an Idealized Computer	19
Memory and Registers inside Intel 32-bit PC.....	20
“Arithmetic” Project: Memory Layout and Registers	21
“Arithmetic” Project: A Computer Program	22
“Arithmetic” Project: Assigning Numbers to Memory Locations	23
Assigning Numbers to Registers	25
“Arithmetic” Project: Adding Numbers to Memory Cells.....	26
Incrementing/Decrementing Numbers in Memory and Registers.....	28
Multiplying Numbers.....	30
Multiplication and Registers.....	32
Chapter x86.2: Debug and Release Binaries.....	33
“Arithmetic” Project: C/C++ Program.....	33
Downloading and Configuring WinDbg Debugger	34
WinDbg Disassembly Output – Debug Executable	36
WinDbg Disassembly Output – Release Executable.....	45
Chapter x86.3: Number Representations.....	46
Numbers and Their Representations.....	46
Decimal Representation (Base Ten).....	47

Ternary Representation (Base Three).....	48
Binary Representation (Base Two)	49
Hexadecimal Representation (Base Sixteen).....	50
Why are Hexadecimals Used?	51
Chapter x86.4: Pointers	53
A Definition	53
“Pointers” Project: Memory Layout and Registers	54
“Pointers” Project: Calculations.....	55
Using Pointers to Assign Numbers to Memory Cells	56
Adding Numbers Using Pointers.....	62
Multiplying Numbers Using Pointers.....	65
Chapter x86.5: Bytes, Words, and Double Words	68
Using Hexadecimal Numbers	68
Byte Granularity.....	69
Bit Granularity	70
Memory Layout.....	71
Chapter x86.6: Pointers to Memory.....	73
Pointers Revisited	73
Addressing Types.....	74
Registers Revisited	80
NULL Pointers.....	81
Invalid Pointers.....	82
Variables as Pointers	83
Pointer Initialization	84
Note: Initialized and Uninitialized Data.....	85
More Pseudo Notation.....	86

“MemoryPointers” Project: Memory Layout.....	87
Chapter x86.7: Logical Instructions and EIP	95
Instruction Format.....	95
Logical Shift Instructions	96
Logical Operations	97
Zeroing Memory or Registers.....	98
Instruction Pointer	99
Note: Code Section	100
Chapter x86.8: Reconstructing a Program with Pointers	102
Example of Disassembly Output: No Optimization	102
Reconstructing C/C++ Code: Part 1	104
Reconstructing C/C++ Code: Part 2	105
Reconstructing C/C++ Code: Part 3	106
Reconstructing C/C++ Code: C/C++ program	107
Example of Disassembly Output: Optimized Program.....	108
Chapter x86.9: Memory and Stacks.....	109
Stack: A Definition.....	109
Stack Implementation in Memory	110
Things to Remember.....	112
PUSH Instruction	113
POP instruction	114
Register Review	115
Application Memory Simplified.....	116
Stack Overflow.....	117
Jumps.....	119
Calls.....	121

Call Stack.....	123
Exploring Stack in WinDbg.....	125
Chapter x86.10: Frame Pointer and Local Variables	128
Stack Usage	128
Register Review	129
Addressing Array Elements	130
Stack Structure (No Function Parameters)	131
Function Prolog.....	132
Raw Stack (No Local Variables and Function Parameters)	133
Function Epilog	135
“Local Variables” Project.....	136
Disassembly of Optimized Executable (Release Configuration).....	139
Advanced Topic: FPO.....	140
Chapter x86.11: Function Parameters.....	141
“FunctionParameters” Project.....	141
Stack Structure	142
Stack Structure with FPO	144
Function Prolog and Epilog.....	145
Project Disassembled Code with Comments.....	146
Release Build with FPO Enabled	149
Cdecl Calling Convention.....	151
Parameter Mismatch Problem	152
Chapter x86.12: More Instructions	153
CPU Flags Register	153
The Fastest Way to Fill Memory.....	154
Testing for 0.....	156

TEST - Logical Compare.....	157
CMP – Compare Two Operands.....	158
TEST or CMP?.....	159
Conditional Jumps.....	160
The Structure of Registers.....	161
Function Return Value	162
Using Byte Registers	163
Chapter x86.13: Function Pointer Parameters.....	164
“FunctionPointerParameters” Project.....	164
Commented Disassembly.....	165
Dynamic Addressing of Local Variables.....	168
Chapter x86.14: Summary of Code Disassembly Patterns	170
Function Prolog/Epilog	170
Passing Parameters	171
LEA (Load Effective Address)	172
Accessing Parameters and Local Variables	173
Appendix x86: Using Docker Environment	174
Chapter x64.1: Memory, Registers, and Simple Arithmetic	176
Memory and Registers inside an Idealized Computer	176
Memory and Registers inside Intel 64-bit PC.....	177
“Arithmetic” Project: Memory Layout and Registers	178
“Arithmetic” Project: A Computer Program	179
“Arithmetic” Project: Assigning Numbers to Memory Locations	180
Assigning Numbers to Registers	182
“Arithmetic” Project: Adding Numbers to Memory Cells.....	183
Incrementing/Decrementing Numbers in Memory and Registers.....	186

Multiplying Numbers.....	189
Chapter x64.2: Debug and Release Binaries.....	191
“Arithmetic” Project: C/C++ Program.....	191
Downloading and Configuring WinDbg Debugger	192
WinDbg Disassembly Output – Debug Executable	194
WinDbg Disassembly Output – Release Executable.....	203
Chapter x64.3: Number Representations.....	204
Numbers and Their Representations.....	204
Decimal Representation (Base Ten).....	205
Ternary Representation (Base Three).....	206
Binary Representation (Base Two)	207
Hexadecimal Representation (Base Sixteen).....	208
Why are Hexadecimals Used?	209
Chapter x64.4: Pointers	211
A Definition	211
“Pointers” Project: Memory Layout and Registers	212
“Pointers” Project: Calculations.....	213
Using Pointers to Assign Numbers to Memory Cells	214
Adding Numbers Using Pointers.....	221
Multiplying Numbers Using Pointers.....	225
Chapter x64.5: Bytes, Words, and Double Words.....	229
Using Hexadecimal Numbers	229
Byte Granularity.....	230
Bit Granularity.....	231
Memory Layout.....	233
Chapter x64.6: Pointers to Memory.....	235

Pointers Revisited	235
Addressing Types	236
Registers Revisited	242
NULL Pointers.....	243
Invalid Pointers.....	244
Variables as Pointers	245
Pointer Initialization	246
Note: Initialized and Uninitialized Data.....	247
More Pseudo Notation.....	248
“MemoryPointers” Project: Memory Layout.....	249
Chapter x64.7: Logical Instructions and EIP	258
Instruction Format.....	258
Logical Shift Instructions	259
Logical Operations	260
Zeroing Memory or Registers.....	261
Instruction Pointer	262
Note: Code Section	264
Chapter x64.8: Reconstructing a Program with Pointers	266
Example of Disassembly Output: No Optimization	266
Reconstructing C/C++ Code: Part 1	269
Reconstructing C/C++ Code: Part 2	271
Reconstructing C/C++ Code: Part 3	272
Reconstructing C/C++ Code: C/C++ program	273
Example of Disassembly Output: Optimized Program.....	274
Chapter x64.9: Memory and Stacks.....	275
Stack: A Definition.....	275

Stack Implementation in Memory	276
Things to Remember	278
PUSH Instruction	279
POP instruction	280
Register Review	281
Application Memory Simplified.....	282
Stack Overflow.....	283
Jumps.....	285
Calls.....	287
Call Stack.....	289
Exploring Stack in WinDbg.....	291
Chapter x64.10: Local Variables.....	295
Stack Usage	295
Addressing Array Elements	296
Stack Structure (No Function Parameters)	297
Function Prolog.....	298
Function Epilog	299
“Local Variables” Project.....	300
Disassembly of Optimized Executable (Release Configuration).....	304
Chapter x64.11: Function Parameters.....	305
“FunctionParameters” Project	305
Stack Structure	306
Function Prolog and Epilog.....	308
Project Disassembled Code with Comments.....	310
Parameter Mismatch Problem	313
Chapter x64.12: More Instructions	314

CPU Flags Register	314
The Fastest Way to Fill Memory.....	315
Testing for 0.....	317
TEST - Logical Compare.....	318
CMP – Compare Two Operands.....	319
TEST or CMP?.....	320
Conditional Jumps.....	321
The Structure of Registers.....	322
Function Return Value	323
Using Byte Registers	324
Chapter x64.13: Function Pointer Parameters.....	325
“FunctionPointerParameters” Project.....	325
Commented Disassembly.....	326
Chapter x64.14: Summary of Code Disassembly Patterns	329
Function Prolog/Epilog	329
Parameters and Local Variables.....	331
LEA (Load Effective Address)	333
Accessing Parameters and Local Variables	334
Appendix x64: Using Docker Environment	335