

# Windows Debugging, Disassembling, Reversing

---

Practical Foundations: Training Course

Dmitry Vostokov  
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

Copyright © 2009 by Dmitry Vostokov

Copyright © 2015 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.

You must not circulate this book in any other binding or cover and you must impose the same condition on any acquirer.

OpenTask books are available through booksellers and distributors worldwide. For further information or comments send requests to:

[press@opentask.com](mailto: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: 978-1-908043-94-8

First printing, 2015

Revision 2.0

## Summary of Contents

Contents.....	5
Preface to the New Edition .....	15
Combined Preface from Previous Editions .....	17
About the Author .....	19
Chapter x86.1: Memory, Registers, and Simple Arithmetic .....	21
Chapter x86.2: Debug and Release Binaries.....	35
Chapter x86.3: Number Representations .....	50
Chapter x86.4: Pointers .....	57
Chapter x86.5: Bytes, Words, and Double Words .....	73
Chapter x86.6: Pointers to Memory.....	78
Chapter x86.7: Logical Instructions and EIP .....	100
Chapter x86.8: Reconstructing a Program with Pointers .....	108
Chapter x86.9: Memory and Stacks.....	116
Chapter x86.10: Frame Pointer and Local Variables .....	136
Chapter x86.11: Function Parameters .....	151
Chapter x86.12: More Instructions .....	165
Chapter x86.13: Function Pointer Parameters.....	176
Chapter x86.14: Summary of Code Disassembly Patterns .....	182
Chapter x64.1: Memory, Registers, and Simple Arithmetic .....	187
Chapter x64.2: Debug and Release Binaries.....	202
Chapter x64.3: Number Representations.....	217
Chapter x64.4: Pointers .....	224
Chapter x64.5: Bytes, Words, and Double Words .....	242
Chapter x64.6: Pointers to Memory.....	248
Chapter x64.7: Logical Instructions and EIP .....	271

Chapter x64.8: Reconstructing a Program with Pointers .....	279
Chapter x64.9: Memory and Stacks.....	288
Chapter x64.10: Local Variables.....	308
Chapter x64.11: Function Parameters.....	320
Chapter x64.12: More Instructions .....	330
Chapter x64.13: Function Pointer Parameters.....	341
Chapter x64.14: Summary of Code Disassembly Patterns .....	345

# Contents

Contents.....	5
Preface to the New Edition .....	15
Combined Preface from Previous Editions.....	17
About the Author .....	19
Chapter x86.1: Memory, Registers, and Simple Arithmetic .....	21
Memory and Registers inside an Idealized Computer .....	21
Memory and Registers inside Intel 32-bit PC.....	22
“Arithmetic” Project: Memory Layout and Registers .....	23
“Arithmetic” Project: A Computer Program .....	24
“Arithmetic” Project: Assigning Numbers to Memory Locations .....	25
Assigning Numbers to Registers .....	27
“Arithmetic” Project: Adding Numbers to Memory Cells.....	28
Incrementing/Decrementing Numbers in Memory and Registers.....	30
Multiplying Numbers.....	32
Multiplication and Registers.....	34
Chapter x86.2: Debug and Release Binaries.....	35
“Arithmetic” Project: C/C++ Program.....	35
Downloading and Configuring WinDbg Debugger .....	36
WinDbg Disassembly Output – Debug Executable .....	38
WinDbg Disassembly Output – Release Executable.....	49
Chapter x86.3: Number Representations.....	50
Numbers and Their Representations.....	50
Decimal Representation (Base Ten).....	51
Ternary Representation (Base Three).....	52

Binary Representation (Base Two) .....	53
Hexadecimal Representation (Base Sixteen).....	54
Why Hexadecimals are used?.....	55
Chapter x86.4: Pointers .....	57
A Definition .....	57
“Pointers” Project: Memory Layout and Registers.....	58
“Pointers” Project: Calculations.....	59
Using Pointers to Assign Numbers to Memory Cells .....	60
Adding Numbers Using Pointers.....	66
Multiplying Numbers Using Pointers.....	69
Chapter x86.5: Bytes, Words, and Double Words .....	73
Using Hexadecimal Numbers .....	73
Byte Granularity.....	74
Bit Granularity .....	75
Memory Layout.....	76
Chapter x86.6: Pointers to Memory.....	78
Pointers Revisited .....	78
Addressing Types.....	79
Registers Revisited .....	85
NULL Pointers.....	86
Invalid Pointers.....	87
Variables as Pointers .....	88
Pointer Initialization .....	89
Note: Initialized and Uninitialized Data.....	90
More Pseudo Notation.....	91
“MemoryPointers” Project: Memory Layout.....	92

Chapter x86.7: Logical Instructions and EIP .....	100
Instruction Format.....	100
Logical Shift Instructions .....	101
Logical Operations .....	102
Zeroing Memory or Registers.....	103
Instruction Pointer .....	104
Note: Code Section.....	105
Chapter x86.8: Reconstructing a Program with Pointers .....	108
Example of Disassembly Output: No Optimization .....	108
Reconstructing C/C++ Code: Part 1 .....	111
Reconstructing C/C++ Code: Part 2 .....	112
Reconstructing C/C++ Code: Part 3 .....	113
Reconstructing C/C++ Code: C/C++ program .....	114
Example of Disassembly Output: Optimized Program.....	115
Chapter x86.9: Memory and Stacks.....	116
Stack: A Definition.....	116
Stack Implementation in Memory.....	117
Things to Remember.....	119
PUSH Instruction .....	120
POP instruction .....	121
Register Review .....	122
Application Memory Simplified.....	123
Stack Overflow.....	124
Jumps.....	126
Calls.....	128
Call Stack.....	130

Exploring Stack in WinDbg.....	132
Chapter x86.10: Frame Pointer and Local Variables .....	136
Stack Usage .....	136
Register Review .....	137
Addressing Array Elements .....	138
Stack Structure (No Function Parameters) .....	139
Raw Stack (No Local Variables and Function Parameters) .....	140
Function Prolog.....	141
Function Epilog .....	142
“Local Variables” Project.....	143
Disassembly of Optimized Executable (Release Configuration).....	148
Advanced Topic: FPO.....	149
Chapter x86.11: Function Parameters.....	151
“FunctionParameters” Project.....	151
Stack Structure .....	152
Stack Structure with FPO .....	154
Function Prolog and Epilog.....	156
Project Disassembled Code with Comments.....	157
Release Build with FPO Enabled.....	162
Cdecl Calling Convention.....	163
Parameter Mismatch Problem .....	164
Chapter x86.12: More Instructions .....	165
CPU Flags Register .....	165
The Fastest Way to Fill Memory.....	166
Testing for 0.....	168
TEST - Logical Compare.....	169



CMP – Compare Two Operands.....	170
TEST or CMP?.....	171
Conditional Jumps.....	172
The Structure of Registers.....	173
Function Return Value.....	174
Using Byte Registers.....	175
Chapter x86.13: Function Pointer Parameters.....	176
“FunctionPointerParameters” Project.....	176
Commented Disassembly.....	177
Dynamic Addressing of Local Variables.....	180
Chapter x86.14: Summary of Code Disassembly Patterns.....	182
Function Prolog / Epilog.....	182
Passing Parameters.....	183
LEA (Load Effective Address).....	184
Accessing Parameters and Local Variables.....	185
Chapter x64.1: Memory, Registers, and Simple Arithmetic.....	187
Memory and Registers inside an Idealized Computer.....	187
Memory and Registers inside Intel 64-bit PC.....	188
“Arithmetic” Project: Memory Layout and Registers.....	189
“Arithmetic” Project: A Computer Program.....	190
“Arithmetic” Project: Assigning Numbers to Memory Locations.....	191
Assigning Numbers to Registers.....	193
“Arithmetic” Project: Adding Numbers to Memory Cells.....	194
Incrementing/Decrementing Numbers in Memory and Registers.....	197
Multiplying Numbers.....	200
Chapter x64.2: Debug and Release Binaries.....	202

“Arithmetic” Project: C/C++ Program.....	202
Downloading and Configuring WinDbg Debugger .....	203
WinDbg Disassembly Output – Debug Executable .....	205
WinDbg Disassembly Output – Release Executable.....	216
Chapter x64.3: Number Representations.....	217
Numbers and Their Representations.....	217
Decimal Representation (Base Ten).....	218
Ternary Representation (Base Three).....	219
Binary Representation (Base Two) .....	220
Hexadecimal Representation (Base Sixteen).....	221
Why Hexadecimals are used?.....	222
Chapter x64.4: Pointers .....	224
A Definition .....	224
“Pointers” Project: Memory Layout and Registers .....	225
“Pointers” Project: Calculations.....	226
Using Pointers to Assign Numbers to Memory Cells .....	227
Adding Numbers Using Pointers.....	234
Multiplying Numbers Using Pointers.....	238
Chapter x64.5: Bytes, Words, and Double Words .....	242
Using Hexadecimal Numbers .....	242
Byte Granularity.....	243
Bit Granularity .....	244
Memory Layout.....	246
Chapter x64.6: Pointers to Memory.....	248
Pointers Revisited .....	248
Addressing Types.....	249

Registers Revisited .....	255
NULL Pointers.....	256
Invalid Pointers.....	257
Variables as Pointers .....	258
Pointer Initialization.....	259
Note: Initialized and Uninitialized Data.....	260
More Pseudo Notation.....	261
“MemoryPointers” Project: Memory Layout.....	262
Chapter x64.7: Logical Instructions and EIP.....	271
Instruction Format.....	271
Logical Shift Instructions .....	272
Logical Operations .....	273
Zeroing Memory or Registers.....	274
Instruction Pointer .....	275
Note: Code Section.....	277
Chapter x64.8: Reconstructing a Program with Pointers .....	279
Example of Disassembly Output: No Optimization .....	279
Reconstructing C/C++ Code: Part 1 .....	282
Reconstructing C/C++ Code: Part 2 .....	284
Reconstructing C/C++ Code: Part 3 .....	285
Reconstructing C/C++ Code: C/C++ program .....	286
Example of Disassembly Output: Optimized Program.....	287
Chapter x64.9: Memory and Stacks.....	288
Stack: A Definition.....	288
Stack Implementation in Memory.....	289
Things to Remember.....	291

PUSH Instruction .....	292
POP instruction .....	293
Register Review .....	294
Application Memory Simplified.....	295
Stack Overflow.....	296
Jumps.....	298
Calls.....	300
Call Stack.....	302
Exploring Stack in WinDbg.....	304
Chapter x64.10: Local Variables.....	308
Stack Usage .....	308
Addressing Array Elements .....	309
Stack Structure (No Function Parameters) .....	310
Function Prolog.....	311
Function Epilog .....	312
“Local Variables” Project.....	313
Disassembly of Optimized Executable (Release Configuration).....	319
Chapter x64.11: Function Parameters.....	320
“FunctionParameters” Project.....	320
Stack Structure .....	321
Function Prolog and Epilog.....	323
Project Disassembled Code with Comments.....	325
Parameter Mismatch Problem .....	329
Chapter x64.12: More Instructions .....	330
CPU Flags Register .....	330
The Fastest Way to Fill Memory.....	331

Testing for 0.....	333
TEST - Logical Compare.....	334
CMP – Compare Two Operands.....	335
TEST or CMP?.....	336
Conditional Jumps.....	337
The Structure of Registers.....	338
Function Return Value.....	339
Using Byte Registers.....	340
Chapter x64.13: Function Pointer Parameters.....	341
“FunctionPointerParameters” Project.....	341
Commented Disassembly.....	342
Chapter x64.14: Summary of Code Disassembly Patterns.....	345
Function Prolog / Epilog.....	345
Parameters and Local Variables.....	347
LEA (Load Effective Address).....	349
Accessing Parameters and Local Variables.....	350