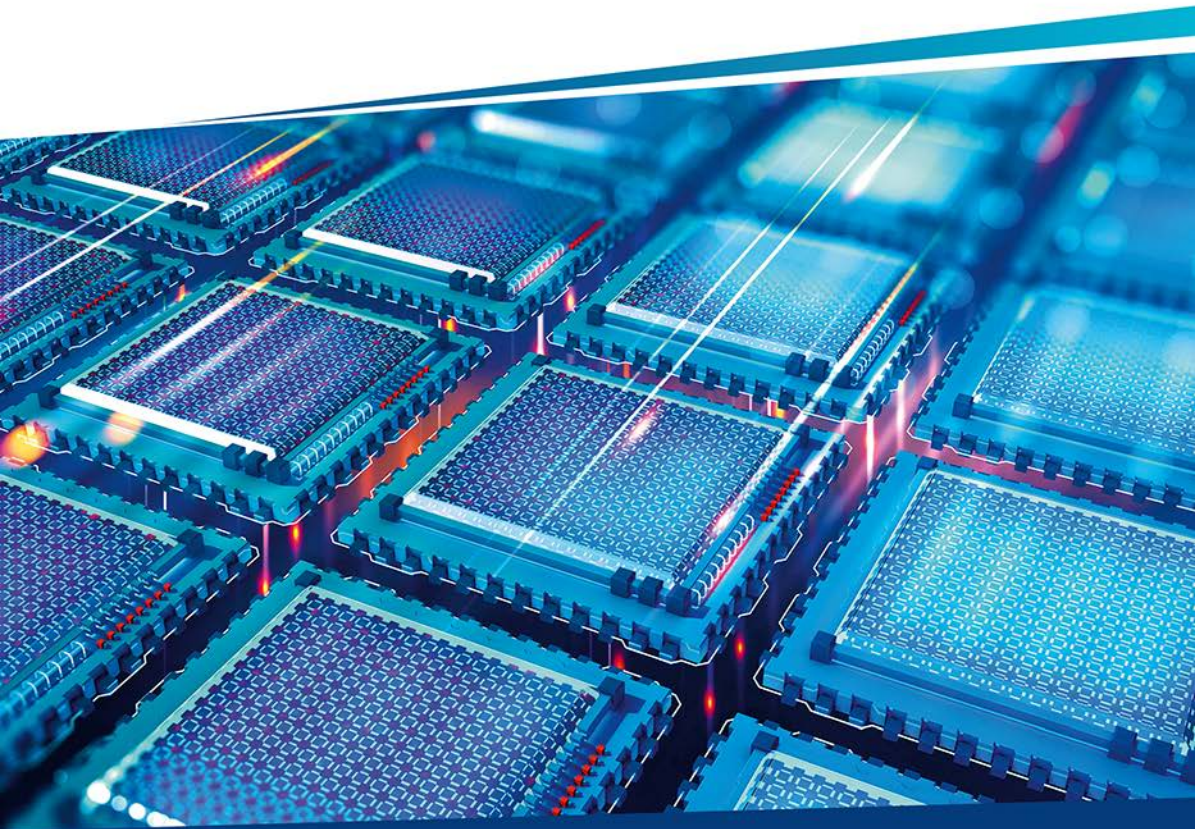


Dawid **Farbaniec**

Asembler

Programowanie



Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/asempr>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-5495-1

Copyright © Helion 2019

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Słowem wstępu11
O książce11
O autorze11

Część I. Wprowadzenie 13

Rozdział 1. Od początku15
1.1. Systemy liczbowe15
1.2. Liczby ze znakiem i bez znaku17
 Wartość rozszerzona z zachowaniem znaku18
1.3. Kod ASCII i Unicode19
1.4. Systemy operacyjne Windows21
1.5. Podstawy działania kompilatorów22

Rozdział 2. Architektura procesorów x86(-64)25
2.1. Tryby pracy25
2.2. Rejestry procesora26
2.3. Pamięć operacyjna31
2.4. Stos33
2.5. Tryby adresowania33
2.6. Kod maszynowy34
2.7. Format kodowania instrukcji34
2.8. Przerwania35
2.9. Zestawy instrukcji35
2.10. Koprocessor35

Rozdział 3. Narzędzia programistyczne37

3.1.	Visual Studio i rozszerzenie do programowania w Asemblerze	37
	Tworzenie projektu MASM w Visual Studio	38
3.2.	Alternatywne podejście — wyodrębnienie narzędzi konsolowych	42
3.3.	Edytor zasobów	44
3.4.	Edytor heksadecymalny	44
3.5.	Debugger	45

Rozdział 4. Wstęp do Asemblera x8647

4.1.	Trochę historii — Asembler dla 16-bitowego podsystemu MS-DOS	47
4.2.	Program „Witaj, 32-bitowy świecie Asemblera!”	51
4.3.	Składnia wysokopoziomowa w Asemblerze	52

Rozdział 5. Wstęp do Asemblera x86-6455

5.1.	Program „Witaj, 64-bitowy świecie Asemblera!”	55
5.2.	Debugger — analiza programu krok po kroku	56
5.3.	Zmienne i stałe	57
	Typy całkowite	57
	Typy zmiennoprzecinkowe	58
	Struktury	58
	Stałe	59
5.4.	Instrukcje skoku warunkowego i bezwarunkowego	59
5.5.	Etykiety nazwane i anonimowe	62
5.6.	Metody tworzenia pętli	63
5.7.	Konwencje wywoływania funkcji	64
5.8.	Tworzenie własnych funkcji (procedur)	65

Część II. Programowanie w Asemblerze x86-64 dla Windows69**Rozdział 6. Podstawy programowania aplikacji Windows71**

6.1.	Tworzenie konsoli tekstowej	71
6.2.	Pobieranie i wyświetlanie danych	71
6.3.	Kolory tła i tekstu w konsoli	74
6.4.	Tworzenie okna dialogowego	76
6.5.	Kontrolki interfejsu graficznego użytkownika	78

Rozdział 7. Napisy (ciągi znaków)	81
7.1. Deklaracja ciągów znaków	81
7.2. Dyrektywa <code>byte ptr []</code> i podobne	81
7.3. Kopiowanie napisów	82
Rozdział 8. Informacje o środowisku pracy	85
8.1. Identyfikacja procesora	85
8.2. Odczytywanie wersji systemu	86
Rozdział 9. Praca z plikami	89
9.1. Tworzenie plików i folderów	89
9.2. Zapis danych do pliku	91
9.3. Odczyt danych z pliku	92
9.4. Ustawianie wskaźnika pliku	93
9.5. Usuwanie plików	94
9.6. Zamykanie uchwytu pliku	95
Rozdział 10. Podstawy wielozadaniowości	97
10.1. Wątki	97
10.2. Procesy	98
10.3. Czasomierz	101
Rozdział 11. Programowanie sieciowe	103
11.1. Obsługa protokołu FTP	103
11.2. Gniazda systemu Windows	105
Funkcje WinSock	105
Aplikacja w architekturze klient – serwer	106
Rozdział 12. Łączenie Asemblera z językiem C++	115
12.1. Funkcje Asemblera w projekcie Visual C++	115
12.2. Funkcje wewnętrzne (ang. <i>intrinsics</i>) w Visual C++	116
Przykład użycia funkcji wewnętrznej (ang. <i>intrinsic</i>)	116

Część III. Asembler x86-64 w inżynierii odwrotnej kodu (RCE) ... 119

Rozdział 13. Narzędzia używane w RCE 121

13.1. Podstawowe narzędzia	121
Odpluskwacz (ang. debugger)	121
Deassembler (ang. disassembler)	122
Edytor szesnastkowy (heksadecymalny)	122
Identyfikator plików	124
Edytor zasobów (ang. resource editor)	124
Dekompilator (ang. decompiler)	125
13.2. Więcej narzędzi — szybsza praca	125

Rozdział 14. Format plików Portable Executable 127

14.1. Ogólna budowa pliku PE	127
14.2. Przeglądanie struktury pliku PE	129

Rozdział 15. Proste metody utrudniające analizę 131

15.1. Szyfrowanie napisów	131
15.2. Wykrywanie narzędzi typu debugger	131
15.3. Dynamiczne wywoływanie funkcji API	132
15.4. Wykrywanie okna określonego narzędzia	133
15.5. Śmieciowy kod (ang. junk code)	134

Rozdział 16. Modyfikacja plików wykonywalnych PE 137

16.1. Modyfikacja kodu w debuggerze	137
16.2. Modyfikacja kodu w edytorze szesnastkowym	138
16.3. Tworzenie programu typu crack	139

Część IV. Inne odmiany języka Asembler 143

Rozdział 17. Wprowadzenie do MSIL/CIL — Asemblera platformy .NET 145

17.1. MSIL/CIL — informacje ogólne	145
17.2. Program „Witaj, świecie!” w Asemblerze MSIL	145
17.3. Dekompilacja — uzyskanie kodu pośredniego z pliku EXE	146
Metody ochrony przed dekompilacją i analizą	147

Rozdział 18. Wprowadzenie do WebAssembly — Asemblera dla aplikacji webowych 149

- 18.1. WebAssembly — informacje ogólne149
- 18.2. Program „Witaj, świecie WebAssembly!”150

Część V. Instrukcje procesorów x86(-64)151

Rozdział 19. Asembler x86(-64) — instrukcje ogólnego przeznaczenia 153

- 19.1. Instrukcje transferu danych153
 - Instrukcja MOV153
 - Instrukcje kopiowania warunkowego CMOVcc154
 - Instrukcja XCHG156
 - Instrukcja BSWAP157
 - Instrukcja XADD158
 - Instrukcja CMPXCHG159
 - Instrukcje CMPXCHG8B/CMPXCHG16B159
 - Instrukcja PUSH160
 - Instrukcja POP161
 - Instrukcje PUSHA/PUSHAD161
 - Instrukcje POPA/POPAD161
 - Instrukcje CWD/CDQ/CQO162
 - Instrukcje CBW/CWDE/CDQE163
 - Instrukcje MOVSX/MOVSXD163
 - Instrukcja MOVZX164
- 19.2. Instrukcje arytmetyczne165
 - Instrukcja ADCX165
 - Instrukcja ADOX166
 - Instrukcja ADD166
 - Instrukcja ADC167
 - Instrukcja SUB167
 - Instrukcja SBB168
 - Instrukcja IMUL169
 - Instrukcja MUL169
 - Instrukcja IDIV170
 - Instrukcja DIV170
 - Instrukcja INC171
 - Instrukcja DEC171
 - Instrukcja NEG171
 - Instrukcja CMP172

19.3.	Instrukcje logiczne	172
	Instrukcja AND	172
	Instrukcja OR	172
	Instrukcja XOR	173
	Instrukcja NOT	173
19.4.	Instrukcje przesunięć i obrotów	174
	Instrukcje SAL/SHL	174
	Instrukcja SAR	174
	Instrukcja SHR	175
	Instrukcja RCL	176
	Instrukcja RCR	177
	Instrukcja ROL	177
	Instrukcja ROR	178
	Instrukcja SHRD	179
	Instrukcja SHLD	180
19.5.	Instrukcje do operacji na bitach i bajtach	181
	Instrukcja BT	181
	Instrukcja BTS	181
	Instrukcja BTR	181
	Instrukcja BTC	182
	Instrukcja BSF	182
	Instrukcja BSR	183
	Instrukcje SETcc	183
	Instrukcja TEST	185
	Instrukcja CRC32	186
	Instrukcja POPCNT	186
19.6.	Instrukcje manipulacji bitowych	187
	Instrukcja ANDN	187
	Instrukcja BEXTR	187
	Instrukcja BLSI	188
	Instrukcja BLSMSK	188
	Instrukcja BLSR	188
	Instrukcja BZHI	189
	Instrukcja LZCNT	189
	Instrukcja MULX	190
	Instrukcja PDEP	190
	Instrukcja PEXT	191
	Instrukcja RORX	191
	Instrukcje SARX, SHLX, SHRX	192
	Instrukcja TZCNT	192

19.7. Instrukcje kontroli przepływu	193
Instrukcja JMP	193
Instrukcje Jcc	193
Instrukcje LOOP/LOOPcc	195
Instrukcja CALL	196
Instrukcja RET	196
19.8. Instrukcje do operacji na napisach	197
Instrukcje MOVS*	197
Instrukcje CMPS*	198
Instrukcje LODS*	199
Instrukcje STOS*	200
Instrukcje SCAS*	200
19.9. Instrukcje wejścia-wyjścia	201
Instrukcja IN	201
Instrukcja OUT	202
Instrukcje INS*	202
Instrukcje OUTS*	202
19.10. Instrukcje kontroli flag	203
19.11. Instrukcje różne	203
Instrukcja LEA	203
Instrukcja NOP	204
Instrukcja UD2	204
Instrukcja CUID	204
Instrukcja MOVBE	205
Zakończenie	207
Skorowidz	209

Rozdział 12.

Łączenie Asemblera z językiem C++

12.1. Funkcje Asemblera w projekcie Visual C++

Po utworzeniu projektu w środowisku Visual Studio dla języka C++ (Visual C++) mamy możliwość pisania funkcji w języku Asembler. Poszczególne kroki tworzenia takiego projektu przedstawiono w podrozdziale 3.1. „Tworzenie projektu MASM w Visual Studio”.

Łączenie programu w Visual C++ z procedurami napisanymi w Asemblerze MASM polega na dodaniu do kodu w C++ zapisu:

```
extern "C" void my_assembler_procedure1();
```

Oczywiście nazwa procedury (`my_assembler_procedure1`) oraz zwracany typ (`void`) powinny być zastąpione zgodnie ze składnią tworzonej procedury.

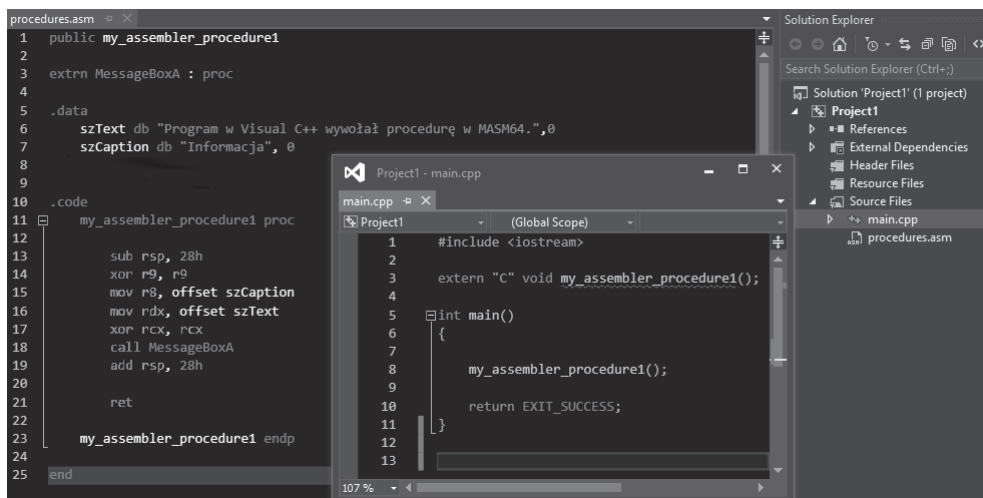
Zapis `extern "C"` powoduje tutaj odwołanie się z pliku źródłowego w C++ (np. *main.cpp*) do procedury zapisanej w języku Asembler (plik o rozszerzeniu *.asm*). Zarówno plik *.cpp*, jak i *.asm* należy standardowo dodać do plików projektu w gałęzi *Source Files* w oknie *Solution Explorer*.

W przypadku niewykrycia funkcji asemblerowej przez konsolidator (ang. *linker*) należy dodać na początku kodu źródłowego w języku Asembler (*.asm*) zapis:

```
public my_assembler_procedure1
```

Zmieni to widoczność określonej procedury lub grupy procedur.

Przykładowy projekt przedstawiono na rysunku 12.1.



RYSUNEK 12.1. Wywołanie procedury w języku Asembler z programu w Visual C++

12.2. Funkcje wewnętrzne (ang. *intrinsics*) w Visual C++

Kompilator Visual C++ ze środowiska Visual Studio nie wspiera wstawek w języku Asembler w postaci *inline assembly* dla architektury 64-bitowej (x86-64). Zamiast tego udostępnione zostały funkcje wewnętrzne kompilatora, nazywane *intrinsics*. Funkcje te są odwzorowywane na określone instrukcje języka Asembler, które nie są standardowo dostępne z poziomu języka C++.

Na przykład: nie możemy użyć rozkazu asemblerowego BSF (ang. *bit scan forward*) w kodzie Visual C++, ale możliwe jest wykonanie tego typu operacji za pomocą funkcji wewnętrznej o nazwie `_BitScanForward()`.

Przykład użycia funkcji wewnętrznej (ang. *intrinsic*)

W poniższym przykładzie wykorzystana została funkcja wewnętrzna `_mul128`, która wykonuje mnożenie dwóch liczb typu `long long` (64 bity) i zwraca 128-bitowy wynik w dwóch częściach po 64 bity (dwa razy `long long`).

Składnia tej funkcji prezentuje się następująco:

```
long long _mul128(
    long long Multiplier,
    long long Multiplicand,
    long long *HighProduct
);
```

Zatem funkcja ta przyjmuje jako argumenty: *mnożnik*, *mnożną* oraz *wskaźnik na starszą część iloczynu*, a zwraca *młodsza część iloczynu*. Przykładowe użycie tej funkcji przedstawiono na listingu 12.1.

LISTING 12.1. Mnożenie dwóch liczb typu `long long` za pomocą funkcji wewnętrznej `_mul128` (Visual C++)

```
#include <iostream>
#include <intrin.h>

#pragma intrinsic(_mul128)

int main()
{
    //mnożnik
    long long multiplier = 0xffffffffffffffffLL;
    //mnożna
    long long multiplicand = 0x7f000000LL;
    //iloczyn (dwie części)
    long long highProduct, lowProduct;

    //wywołanie funkcji wewnętrznej (ang. intrinsic)
    lowProduct = _mul128(multiplier, multiplicand, &highProduct);

    //wyświetlenie wyniku na ekranie konsoli
    printf_s("0x%llx * 0x%llx = 0x%llx%llx\n",
        multiplier, multiplicand, highProduct, lowProduct);

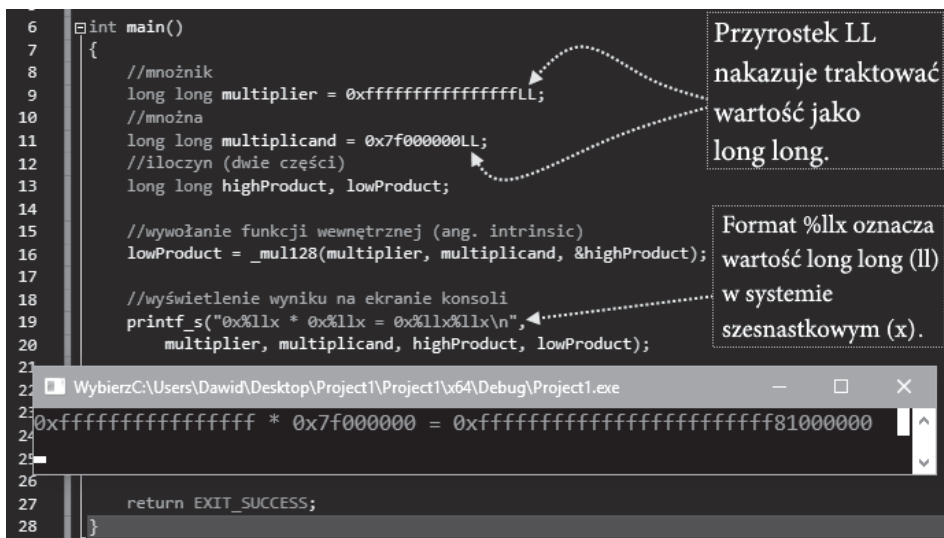
    //zatrzymanie konsoli, ale tylko w trybie DEBUG
    #if _DEBUG
        std::cin.get();
    #endif

    return EXIT_SUCCESS;
}
```

Działanie programu z listingu 12.1 przedstawiono na rysunku 12.2.

Dokumentacja opisująca funkcje wewnętrzne kompilatora Visual C++ dostępna jest pod adresem:

<https://docs.microsoft.com/pl-pl/cpp/intrinsics/compiler-intrinsics>.



RYSUNEK 12.2. Działanie przykładowego programu dla funkcji wewnętrznej `_mul128` (Visual C++)

Skorowidz

A

adresowanie 33, 34
akumulator, 26, 159, 162, 170, 200, *Patrz też:*
 rejestr AX, EAX, EDX
algorytm wzajemnego wykluczania, 98
aplikacja
 instancja, *Patrz:* proces
 klient – serwer, 106
 konsolowa, 71, 72
 ochrona przed analizą, 147
 sieciowa, 105
 webowa, 147
architektura klient – serwer, 106
assembler, 34
Assembler platformy .NET, 145
AsmDude, 37

B

bajt, 17
biblioteka
 kernel32.lib, 43
 User32.lib, 43
bit znaku, 17
błąd, 56
break-point, *Patrz:* pułapka

C

Carry Flag, *Patrz:* flaga CF
Central Processing Unit, *Patrz:* CPU
ciąg znaków, 81
CPU, 35

D

dane
 odczyt, 92
 typ, 19
 zapis, 91
deassembler, 122
debugger, 45, 56, 121
 modyfikacja kodu, 137
 x64dbg, 45, 56, 121, 137
dekompilator, 125
 ILSpy, 125, 146
depacker, 126
Direction Flag, *Patrz:* flaga DF
dyrektywa
 BYTE, 81
 byte ptr, 81
 db, 81
 dword ptr, 82
 ENDP, 65
 ENDPROLOG, 67
 FRAME, 67
 PROC, 65
 PUSHREG, 67
 qword ptr, 82
 word ptr, 82
dysk wirtualny, 49

E

edytor
 heksadecymalny, 44, 122
 modyfikacja kodu, 138
 szesnastkowy, *Patrz:* edytor
 heksadecymalny
 XVI32, 44
 zasobów, 44, 124

emulator DOSBox, 47, 49
Entry Point, *Patrz*: punkt wejścia
etykieta, 59, 62

F

flaga

AF, 159
CF, 31, 60, 61, 159, 165, 166, 167, 170, 171,
172, 173, 174, 175, 176, 177, 178, 179, 180,
181, 182, 186
DF, 31, 197
kierunku, *Patrz*: flaga DF
OF, 31, 61, 159, 166, 170, 172, 173, 174, 176,
178
parzystości, *Patrz*: flaga PF
PF, 31, 62, 159
przeniesienia, *Patrz*: flaga CF
przepełnienia, *Patrz*: flaga OF
SF, 31, 60, 61, 62, 159, 166, 187
zerowa, *Patrz*: flaga ZF
ZF, 31, 60, 61, 62, 64, 159, 160, 182, 183, 187
znaku, *Patrz*: flaga SF

Floating-Point Unit, *Patrz*: koprocesor

folder, 89

format

kodowania instrukcji, 34
PE, 127, 128, 129

FPU, *Patrz*: koprocesor

funkcja

accept, 105
bind, 105
CloseHandle, 95
closesocket, 105
connect, 105
CreateDirectory, 89
CreateFile, 89
CreateProcess, 99
CreateThread, 97
CreateWindowEx, 76
DeleteFile, 94
DialogBoxParam, 76
FtpGetFile, 103
FtpPutFile, 103
FtpSetCurrentDirectory, 103
GetDlgItemInt, 80
GetDlgItemText, 80
GetForegroundWindow, 133

GetProcAddress, 132
GetStdHandle, 71
GetVersion, 86
GetWindowText, 133
InternetConnect, 103
InternetOpen, 103
IsDebuggerPresent, 131
listen, 105
LoadLibrary, 132
Main, 55
MessageBox, 64
ReadConsole, 72
ReadFile, 92, 93
recv, 105
send, 105
SetConsoleTextAttribute, 74
SetDlgItemInt, 80
SetDlgItemText, 80
SetFilePointer, 93, 94
SetTimer, 101
socket, 105
StringCchCopy, 82
WaitForSingleObject, 98
wewnętrzna kompilatora, 116
WinSock, 105
WriteConsole, 71
WriteFile, 91, 92
WSACleanup, 105
WSAStartup, 105
wywołanie
dynamiczne, 132
konwencja, *Patrz*: konwencja

G

gniazdo sieciowe, 105

I

instrukcja

ADC, 167
ADCX, 165, 166
ADD, 166, 167
ADOX, 166
AND, 172
ANDN, 187
arytmetyczna, 165
BEXTR, 187

BLSI, 188
BLSMSK, 188
BLSR, 188
BSF, 182
BSR, 183
BSWAP, 157
BT, 181
BTC, 182
BTR, 181
BTS, 181
BZHI, 189
CALL, 60, 65, 196
CBW, 163
CDQ, 162
CDQE, 163
CLC, 203
CLD, 203
CLI, 203
CMC, 203
CMOV, 154
CMP, 63, 172
CMPS, 198
CMPXCHG, 159
CMPXCHG16B, 159
CMPXCHG8B, 159
CUID, 85, 204
CQO, 162
CRC32, 186
CWD, 162
CWDE, 163
DEC, 171
DIV, 170
do operacji na bitach i bajtach, 181
format kodowania, 34
IDIV, 170
IMUL, 169
IN, 201
INC, 171
INS, 202
J, 193
JAE, 60, 193
JB, 60, 193
JBE, 60, 193
JC, 60, 193
JE, 60, 193
JG, 60, 194
JGE, 60, 194
JL, 60, 194
JLE, 61, 194
JMP, 59, 60, 193
JNA, 61, 194
JNAE, 61, 194
JNB, 61, 194
JNBE, 61, 194
JNC, 61, 194
JNE, 61, 194
JNG, 61, 194
JNGE, 61, 194
JNL, 61, 194
JNLE, 61, 195
JNO, 61, 195
JNP, 61, 195
JNS, 61, 195
JNZ, 61, 63, 195
JO, 62, 195
JP, 62, 195
JPE, 62, 195
JPO, 62, 195
JS, 62, 195
JZ, 60, 62, 194, 195
kontroli flagi, 203
kopiowania warunkowego, 154
LAHF, 203
LEA, 203
LODS, 199
logiczna, 172
LOOP, 60, 63, 195, *Patrz też:* pętla
LOOPE, 196
LOOPNE, 196
LOOPNZ, 64, 196
LOOPZ, 64, 196
LZCNT, 189
manipulacji bitowych, 187
MOV, 82, 153
MOVBE, 205
MOVS, 197
MOVSB, 163
MOVSD, 163
MOVZX, 164
MUL, 169
MULX, 190
NEG, 171
NOP, 204
NOT, 173
OR, 172
OUT, 202

instrukcja

- OUTS, 202
- PDEP, 190
- PEXT, 191
- pętli, *Patrz:* pętla
- POP, 161
- POPA, 161
- POPAD, 161
- POPCNT, 186
- porównania, 63
- PUSH, 160
- PUSHA, 161
- PUSHAD, 161
- PUSHF, 203
- PUSHFD, 203
- PUSHFQ, 203
- RCL, 176
- RCR, 177
- RET, 196
- ROL, 177
- ROR, 178
- RORX, 191
- SAHF, 203
- SAL, 174
- SAR, 174
- SARX, 192
- SBB, 168
- SCAS, 200
- SET, 183
- SHL, 174
- SHLD, 180
- SHLX, 192
- SHR, 175
- SHRD, 179
- SHRX, 192
- skoku
 - bezw warunkowego, 59
 - warunkowego, 59, 60, 61, 63
- STC, 203
- STD, 203
- STI, 203
- STOS, 200
- STOSB, 82
- SUB, 167
- SUB RSP, 65
- TEST, 185
- TZCNT, 192
- UD2, 204

- wejścia-wyjścia, 201

- XADD, 158

- XCHG, 156

- XOR, 173

- interfejs Windows API, 35

- interpreter, 22

- interrupt, *Patrz:* przerwanie

- intrinsic, *Patrz:* funkcja wewnętrzna kompilatora

- inżynieria odwrotna kodu, *Patrz:* RCE

J

jednostka

- główna, *Patrz:* CPU

- zmiennoprzecinkowa, *Patrz:* koprocessor

język

- Asembler, 34

- rejestr, *Patrz:* rejestr

- assembler MASM, 38, 39, 42, 47, 52

- Asembler x86, 47

- CIL, 145

- MSIL, 145

- skryptowy, 22

- WebAssembly, 149

K

kalkulator, 16

kod

- ANSI, 20

- ASCII, 20

- ISO, 20

- maszynowy, 34

- modyfikacja, 139

- w debuggerze, 137

- w edytorze heksadecymalnym, 138

- operacji, *Patrz:* opkod

- pośredni, 22

- strona kodowa, 20

- Unicode, 20

- UTF-8, 21

- wynikowy, 22

- zaciemnianie, 125, 134, 147

kodowanie, 18

- standard, *Patrz:* kod

- U1, 18

- U2, 18

uzupełnienie

do dwóch, *Patrz:* kodowanie U2

do jeden, *Patrz:* kodowanie U1

ZM, 18

znak moduł, *Patrz:* kodowanie ZM

znaków, *Patrz:* kod

kompilator, 22, 34, 43

Emscripten, 150

komunikat, 80

konsola, 71

tekst, 76

kolor, 74

konsolidator, 34

konwencja

Microsoft x64, 64, 65

stdcall, 64

koprocesor, 35

L

liczba

binarna, 16, 17

kodowanie, *Patrz:* kodowanie

ujemna, 17

ze znakiem, 17, 18, 166, 167

heksadecymalna, 16

rozszerzenie, 18

LIFO, 33

linker, *Patrz:* konsolidator

Long mode, *Patrz:* tryb procesora 64-bitowy

M

malware, *Patrz:* oprogramowanie złośliwe

Microsoft Windows, 21

jądro, 21

kalkulator, *Patrz:* kalkulator

Windows 2000, 21

Windows NT, 21, 22

mnemonik, 34, 37

model pamięci

płaski, 32

segmentowy, 31

mutex, 98

mutual exclusion, *Patrz:* algorytm wzajemnego

wykluczania, mutex

O

okno

dialogowe, 76, 78, 80

dynamiczne, 76

konsoli, *Patrz:* konsola

wykrywanie, 133

operand, 34

opkod, 34, 139

oprogramowanie złośliwe, 21

Overflow Flag, *Patrz:* flaga OF

P

pakiet MASM32, 47

pamięć

model, *Patrz:* model pamięci

operacyjna, 31

Parity Flag, *Patrz:* flaga PF

pętla, 63, 196, *Patrz też:* instrukcja LOOP

plik

.rc, 76

.res, 76

binarny, 34

charmap.exe, 21

cvtres.exe, 43

format

PE, *Patrz:* format PE

PE32+, 127

identyfikator, 124

nazwa, 89

odczyt danych, *Patrz:* dane odczyt

pobieranie, 103

prawa

do operacji, 89

dostępu, 89, 90

prog1.asm, 43

resources.res, 43

spakowany, 124, 126

szyfrowanie, 124, 126

napisów, 131

tworzenie, 89, 90

uchwyt, 95

usuwanie, 94

wskaźnik, 93

wykonywalny, 127, *Patrz też:* kod

.exe, 22

plik

- wysyłanie, 103
- zapis danych, *Patrz*: dane zapis
- źródłowy w C++, 115

procedura

- definicja, 66
- epilog, 66
- powrót, 196
- prolog, 66

proces, 98, 99

procesor, 85, 204

- marka, 85
- tryb pracy, *Patrz*: tryb procesora

program

- ConfuserEx, 147
- Detect It Easy, 124
- IDA, 122
- MS-DOS Stub, 128
- PEview, 129
- PUA, 21
- typu
 - crack, 139
 - protector, 147

Protected mode, *Patrz*: tryb procesora

chroniony

protokół FTP, 103

przeniesienie, 60, 193

przepelnienie, 61, 166, 167, 195

przerwanie, 35

pułapka, 45

punkt wejścia, 41

EBP, 161, 162

EBX, 161, 162

ECX, 63, 161, 162

EDI, 161, 162

EDX, 64, 159, 161, 162

ESI, 161, 162

ESP, 161, 162

indeksowy, 28

licznik, *Patrz*: rejestr RCX

nieulotny, 66

R12, 65

R13, 65

R14, 65

R15, 65

R8, 64

R9, 64

RAX, 26, 64, 135, 159, 162, 163, 170, *Patrz też*: akumulatorRBP, 29, 65, *Patrz też*: wskaźnik bazowyRBX, 27, 65, *Patrz też*: rejestr bazowy

RCX, 27, 63, 64

RDI, 29, *Patrz też*: rejestr indeksowyRDX, 28, 64, 162, 170, *Patrz też*: rejestr danychRIP, 30, *Patrz też*: wskaźnik instrukcjiRSI, 29, *Patrz też*: rejestr indeksowyRSP, 30, *Patrz też*: wskaźnik stosu, stos segmentowy, 30, *Patrz też*: flaga

SI, 162

SP, 161, 162

rejestr RSI, 161

Reverse Code Engineering, *Patrz*: RCE**R**

RCE, 121

Real mode, *Patrz*: tryb procesora rzeczywisty

rejestr, 81

- akumulator, *Patrz*: akumulator
- AL, 159
- AX, 159, 161, 162, 163
- bazowy, 26
- BP, 161, 162
- BX, 161, 162
- CX, 63, 161, 162
- danych, 27
- DI, 161, 162
- DX, 161, 162
- EAX, 64, 159, 161, 162, 163

S

sekcja

.code, 55

.const, 55

.data, 55

serwer FTP, 103

Sign Flag, *Patrz*: flaga SF

stała, 55

definicja, 59

standard kodowania, *Patrz*: kod

stos, 33, 64, 162

wyrównanie ręczne, 64

struktura, 58

system

- liczbowy, 15
 - addytywno-subtraktywny, 15
- binarny, 16
- dziesiętny, 16
- heksadecymalny, 16
- pozycyjny, 15, 16
- szesnastkowy, *Patrz:* system liczbowy heksadecymalny
- Microsoft Windows, *Patrz:* Microsoft Windows

T

tryb

- adresowania, *Patrz:* adresowanie procesora, 25

typ, 19

V

- Visual C++, 38, 39, 40, 115, 116
- Visual Studio AsmDude, *Patrz:* AsmDude

W

- wątek, 97
- wielozadaniowość, 97
- wskaźnik
 - bazowy, 29
 - instrukcji, 30
 - stosu, 29, *Patrz też:* stos
- wyjątek
 - Invalid Opcode, 204
 - obsługa, 67
- wykrywanie narzędzi typu debugger, 131

Z

- zdarzenie systemowe, 35
- Zero Flag, *Patrz:* flaga ZF
- zmienna, 57
- znak
 - @@, 62
 - @b, 62
 - @f, 62
 - kodowanie, *Patrz:* kod

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion

Asembler naprawdę jest dla każdego!

- Masz zamiar nauczyć się programować niskopoziomowo?
- Chcesz poznać język Asembler dla architektury x86-64?
- Pragniesz wykorzystać przydatne narzędzia programistyczne?

Jeśli chociaż na jedno z powyższych pytań odpowiedziałeś twierdząco, jesteś na dobrej drodze! Książka *Asembler. Programowanie* bezboleśnie i szybko wprowadzi Cię w tajniki programowania niskopoziomowego, nauczy instrukcji języka Asembler, przedstawi podstawowe informacje o sposobach zapisu liczb i kodowania znaków, zaprezentuje architekturę x86-64 i zasadę działania kompilatorów, a także pokaże narzędzia programistyczne, które z pewnością wzbogacą Twój warsztat.

Dzięki lekturze dowiesz się, jak za pomocą Asemblera tworzyć aplikacje działające w systemach Windows, jak korzystać w nich z danych tekstowych, plików i usług sieciowych, a także jak skutecznie łączyć je z programami napisanymi w języku C++. Poznasz też podstawy Asemblera MSIL/CIL i WebAssembly, jak również sposoby wykorzystania tego języka w inżynierii odwrótej kodu oraz narzędzia, które mogą się do tego przydać. Całość uzupełnia wyczerpujący opis instrukcji procesorów x86(-64) wraz z praktycznymi przykładami ich zastosowania.

- Systemy liczbowe i kodowanie znaków
- Podstawy działania kompilatorów
- Architektura procesorów x86(-64)
- Przegląd narzędzi programistycznych
- Podstawy Asemblera x86 i x86-64
- Zmienne, stałe, typy i struktury
- Instrukcje sterujące i wywoływanie funkcji
- Korzystanie z konsoli i okien dialogowych
- Zastosowanie danych tekstowych
- Użycie plików i funkcji sieciowych
- Łączenie Asemblera z kodem C++
- Asembler i inżynieria odwrótna kodu
- Podstawy Asemblera platformy .NET
- Podstawy WebAssembly
- Opis instrukcji procesorów x86(-64)

Rozwiń skrzydła! Naucz się Asemblera!

  helion.pl  HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	<i>Sprawdź nasze szkolenia!</i>  SZKOLENIA AKADEMIA IT & BUSINESS WWW.SZKOLENIA.HELION.PL	KOD KORZYŚCI <i>Sięgnij po więcej!</i>   ISBN 978-83-283-5495-1  9 788328 354951 INFORMATYKA W NAJLEPSZYM WYDANIU Cena: 44,90 zł
--	---	---