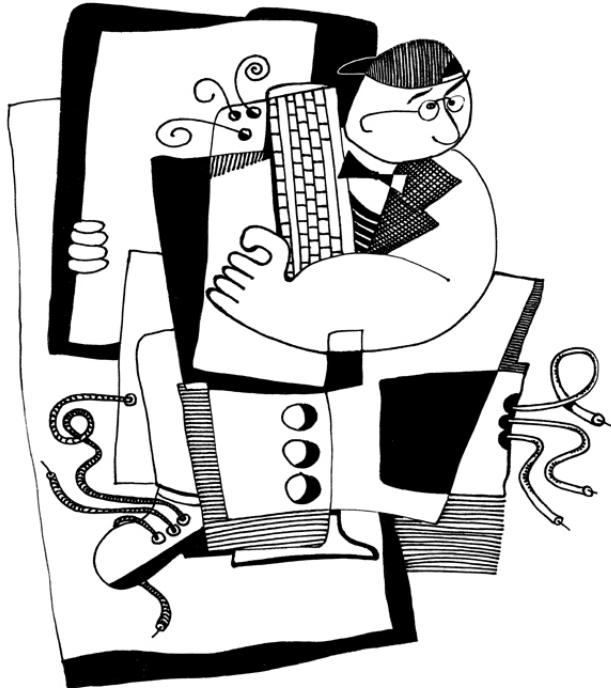


Reverse Engineering para principiantes



Denis Yurichev

Ingeniería inversa para principiantes

Denis Yurichev
<[dennis\(a\)yurichev.com](mailto:dennis(a)yurichev.com)>



©2013-2015, Denis Yurichev.

Esta obra está bajo la licencia Creative Commons «Atribución-NoComercial-SinDerivadas» 3.0 Unported. Para consultar una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-nd/3.0/>.

Versión de este texto (9 de enero de 2026).

La versión más reciente de este texto (y la edición en ruso) está disponible en beginners.re. La versión en formato A4 también está disponible allí.

Existe también una versión LITE (versión introductoria abreviada), pensada para quienes quieren una introducción rápida a los conceptos básicos de la ingeniería inversa: beginners.re

También puedes seguirme en Twitter para enterarte de las actualizaciones de este texto: [@yurichev](https://twitter.com/yurichev)¹ o suscribirte a la lista de correo².

La portada fue realizada por Andy Nechaevsky: [facebook](https://facebook.com/yurichev).

¹twitter.com/yurichev

²yurichev.com

Contenido abreviado

I	Patrones de código	1
II		582
III		591
IV	Java	792
V		840
VI		868
VII	Herramientas	941
VIII		948
IX	Ejemplos de ingeniería inversa de formatos de archivo propie-	

tarios**1107****X****1140****XI****1163****XII Ejercicios****1167****1219****1221****Lista de acrónimos usados****1274**

Índice general

I Patrones de código	1
1 Una breve introducción a la CPU	4
1.1 Algunas palabras sobre diferentes ISA ³ s	5
2 La función más simple	7
2.1 x86	7
2.2 ARM	8
2.3 MIPS	8
2.3.1 Una nota sobre los nombres de instrucciones/registros en MIPS	9
3 Hello, world!	10
3.1 x86	10
3.1.1 MSVC	10
3.1.2 GCC	12
3.1.3 GCC: Sintaxis AT&T	12
3.2 x86-64	14
3.2.1 MSVCx86-64	14
3.2.2 GCCx86-64	14
3.3 GCC	15
3.4 ARM	17
3.4.1 Sin optimización Keil 6/2013 (Modo ARM)	17
3.4.2 Sin optimización Keil 6/2013 (Modo Thumb)	19
3.4.3 Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)	20
3.4.4 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)	21
3.4.5 ARM64	23
3.5 MIPS	25
3.5.1	25

³Instruction Set Architecture

3.5.2	Con optimización GCC	25
3.5.3	Sin optimización GCC	27
3.5.4	29
3.5.5	Con optimización GCC: GDB	29
3.6	Conclusión	30
3.7	Ejercicios	30
3.7.1	Ejercicio #1	30
3.7.2	Ejercicio #2	31
4		32
4.1	Recursión	32
5	Pila	33
5.1	33
5.2	34
5.2.1	34
5.2.2	35
5.2.3	36
5.2.4	x86:	36
5.2.5	(Windows) SEH	38
5.2.6	38
5.2.7	39
5.3	39
5.4	39
5.5	Ejercicios	43
5.5.1	Ejercicio #1	43
5.5.2	Ejercicio #2	43
6	printf() con varios argumentos	46
6.1	x86	46
6.1.1	x86:	46
6.1.2	x64:	55
6.2	ARM	59
6.2.1	ARM:	59
6.2.2	ARM:	61
6.3	MIPS	66
6.3.1	3	66
6.3.2	8	69
6.4	Conclusión	75
6.5	76
7	scanf()	77
7.1	77
7.1.1	77

7.1.2	x86	77
7.1.3	MSVC + OllyDbg	80
7.1.4	x64	83
7.1.5	ARM	84
7.1.6	MIPS	85
7.2		88
7.2.1	MSVC: x86	88
7.2.2	MSVC: x86 + OllyDbg	91
7.2.3	GCC: x86	92
7.2.4	MSVC: x64	92
7.2.5	ARM: Con optimización Keil 6/2013 (Modo Thumb)	93
7.2.6	ARM64	94
7.2.7	MIPS	95
7.3		99
7.3.1	MSVC: x86	100
7.3.2	MSVC: x86: IDA	101
7.3.3	MSVC: x86 + OllyDbg	105
7.3.4	MSVC: x86 + Hiew	107
7.3.5	MSVC: x64	108
7.3.6	ARM	109
7.3.7	MIPS	111
7.3.8	Ejercicio	112
7.4	Ejercicios	112
7.4.1	Ejercicio #1	112
8		114
8.1	x86	114
8.1.1	MSVC	114
8.1.2	MSVC + OllyDbg	115
8.1.3	GCC	116
8.2	x64	117
8.2.1	MSVC	117
8.2.2	GCC	119
8.2.3	GCC: uint64_t int	120
8.3	ARM	121
8.3.1	Sin optimización Keil 6/2013 (Modo ARM)	121
8.3.2	Con optimización Keil 6/2013 (Modo ARM)	122
8.3.3	Con optimización Keil 6/2013 (Modo Thumb)	122
8.3.4	ARM64	122
8.4	MIPS	124
9		126
9.1		126
9.2		127

	ÍNDICE GENERAL
9.3	127
10	130
10.1	130
10.2	136
10.3 Conclusión	140
11	141
11.1	144
11.2 Ejercicio	145
12	146
12.1	146
12.1.1 x86	147
12.1.2 ARM	158
12.1.3 MIPS	161
12.2	165
12.2.1 Con optimización MSVC	166
12.2.2 Con optimización Keil 6/2013: Modo Thumb	166
12.2.3 Con optimización Keil 6/2013: Modo ARM	166
12.2.4 Sin optimización GCC 4.9 (ARM64)	167
12.2.5 MIPS	167
12.2.6 ?	167
12.3	168
12.3.1 x86	168
12.3.2 ARM	169
12.3.3 ARM64	170
12.3.4 MIPS	171
12.3.5	171
12.3.6 Conclusión	172
12.4	172
12.4.1 32-bit	172
12.4.2 64-bit	176
12.4.3 MIPS	178
12.5 Conclusión	179
12.5.1 x86	179
12.5.2 ARM	179
12.5.3 MIPS	179
12.5.4	180
12.6 Ejercicio	181
13 switch()/case/default	182
13.1	182
13.1.1 x86	182

13.1.2 ARM: Con optimización Keil 6/2013 (Modo ARM)	192
13.1.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)	193
13.1.4 ARM64: Sin optimización GCC (Linaro) 4.9	193
13.1.5 ARM64: Con optimización GCC (Linaro) 4.9	194
13.1.6 MIPS	195
13.1.7 Conclusión	196
13.2	196
13.2.1 x86	197
13.2.2 ARM: Con optimización Keil 6/2013 (Modo ARM)	203
13.2.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)	205
13.2.4 MIPS	207
13.2.5 Conclusión	209
13.3	210
13.3.1 MSVC	211
13.3.2 GCC	213
13.3.3 ARM64: Con optimización GCC 4.9.1	213
13.4 Fall-through	215
13.4.1 MSVC x86	216
13.4.2 ARM64	217
13.5 Ejercicios	218
13.5.1 Ejercicio #1	218
14 Lazos	219
14.1	219
14.1.1 x86	219
14.1.2 x86: OllyDbg	224
14.1.3 x86: tracer	224
14.1.4 ARM	227
14.1.5 MIPS	231
14.1.6	232
14.2	233
14.2.1	233
14.2.2 ARM	234
14.2.3 MIPS	235
14.2.4	236
14.3 Conclusión	236
14.4 Ejercicios	238
14.4.1 Ejercicio #1	238
14.4.2 Ejercicio #2	238
14.4.3 Ejercicio #3	239
14.4.4 Ejercicio #4	241
15 Procesamiento de strings C simples	245
15.1 strlen()	245

15.1.1 x86	245
15.1.2 ARM	252
15.1.3 MIPS	255
15.2 Ejercicios	256
15.2.1 Ejercicio #1	256
16 Substitución de instrucciones aritméticas por otros	260
16.1	260
16.1.1	260
16.1.2	261
16.1.3	262
16.2	267
16.2.1	267
16.3 Ejercicios	268
16.3.1 Ejercicio #2	268
17 Unidad de Punto flotante	270
17.1 IEEE 754	270
17.2 x86	270
17.3 ARM, MIPS, x86/x64 SIMD	270
17.4 C/C++	270
17.5	270
17.5.1 x86	271
17.5.2 ARM: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)	278
17.5.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)	279
17.5.4 ARM64: Con optimización GCC (Linaro) 4.9	279
17.5.5 ARM64: Sin optimización GCC (Linaro) 4.9	280
17.5.6 MIPS	281
17.6	282
17.6.1 x86	282
17.6.2 ARM + Sin optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)	283
17.6.3 ARM + Sin optimización Keil 6/2013 (Modo ARM)	284
17.6.4 ARM64 + Con optimización GCC (Linaro) 4.9	285
17.6.5 MIPS	285
17.7	287
17.7.1 x86	287
17.7.2 ARM	317
17.7.3 ARM64	321
17.7.4 MIPS	322
17.8	323
17.9 x64	323
17.10 Ejercicios	323
17.10.1 Ejercicio #1	323
17.10.2 Ejercicio #2	323

18 Matriz	326
18.1	326
18.1.1 x86	326
18.1.2 ARM	329
18.1.3 MIPS	333
18.2	334
18.2.1	334
18.2.2	338
18.3	343
18.3.1 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)	345
18.4	347
18.5	348
18.5.1 x64	348
18.5.2 32-ARM	350
18.5.3 ARM64	351
18.5.4 MIPS	352
18.5.5	353
18.6	356
18.6.1	357
18.6.2	358
18.6.3	361
18.6.4	365
18.7	365
18.7.1 32-bit ARM	367
18.7.2 ARM64	368
18.7.3 MIPS	368
18.7.4 Conclusión	369
18.8 Conclusión	369
18.9 Ejercicios	369
18.9.1 Ejercicio #1	369
18.9.2 Ejercicio #2	374
18.9.3 Ejercicio #3	380
18.9.4 Ejercicio #4	381
18.9.5 Ejercicio #5	383
19 Manipulando bit(s) específicas	389
19.1	389
19.1.1 x86	389
19.1.2 ARM	392
19.2	394
19.2.1 x86	394
19.2.2 ARM + Con optimización Keil 6/2013 (Modo ARM)	401
19.2.3 ARM + Con optimización Keil 6/2013 (Modo Thumb)	401
19.2.4 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)	401

19.2.5 ARM:	402
19.2.6 ARM64: Con optimización GCC (Linaro) 4.9	402
19.2.7 ARM64: Sin optimización GCC (Linaro) 4.9	402
19.2.8 MIPS	403
19.3 Shifts	403
19.4	403
19.4.1	404
19.4.2 x86	404
19.4.3 MIPS	406
19.4.4 ARM	407
19.5	409
19.5.1 x86	411
19.5.2 x64	419
19.5.3 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)	423
19.5.4 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)	423
19.5.5 ARM64 + Con optimización GCC 4.9	424
19.5.6 ARM64 + Sin optimización GCC 4.9	424
19.5.7 MIPS	425
19.6 Conclusión	427
19.6.1	427
19.6.2	428
19.6.3	429
19.6.4	429
19.6.5	429
19.6.6	430
19.7 Ejercicios	430
19.7.1 Ejercicio #1	430
19.7.2 Ejercicio #2	432
19.7.3 Ejercicio #3	435
19.7.4 Ejercicio #4	436
20	439
20.1 x86	439
20.2 x64	441
20.3 32-bit ARM	441
20.4 MIPS	442
20.4.1	444
20.5	445
21 Estructuras	446
21.1 MSVC:	446
21.1.1 OllyDbg	448
21.1.2	449
21.2	450

21.3	UNIX: struct tm	452
21.3.1	Linux	452
21.3.2	ARM	456
21.3.3	MIPS	458
21.3.4	460
21.3.5	462
21.3.6	464
21.4	Organización de campos en la estructura	466
21.4.1	x86	467
21.4.2	ARM	471
21.4.3	MIPS	472
21.4.4	473
21.5	474
21.5.1	OllyDbg	477
21.6	477
21.6.1	477
21.6.2	Trabajando con el tipo float como una estructura	483
21.7	Ejercicios	486
21.7.1	Ejercicio #1	486
21.7.2	Ejercicio #2	487
22		494
22.1	494
22.1.1	x86	495
22.1.2	MIPS	496
22.1.3	ARM (Modo ARM)	498
22.2	500
22.2.1	x86	501
22.2.2	ARM64	501
22.2.3	MIPS	502
22.2.4	Conclusión	502
22.3	502
23		504
23.1	MSVC	505
23.1.1	MSVC + OllyDbg	508
23.1.2	MSVC + tracer	510
23.1.3	MSVC + tracer (code coverage)	512
23.2	GCC	512
23.2.1	GCC + GDB ()	514
23.2.2	GCC + GDB ()	515
24		519
24.1	519

24.1.1 x86	519
24.1.2 ARM	519
24.1.3 MIPS	520
24.2	520
24.2.1 x86	521
24.2.2 ARM	522
24.2.3 MIPS	523
24.3	525
24.3.1 x86	525
24.3.2 ARM	527
24.3.3 MIPS	528
24.4	530
24.4.1 x86	530
24.4.2 ARM	530
24.4.3 MIPS	531
24.5	531
24.5.1 x86	531
24.5.2 ARM	532
24.5.3 MIPS	532
25 SIMD	533
25.1	533
25.1.1	533
25.1.2	540
25.2	546
26	550
26.1 x86-64	550
26.2 ARM	559
26.3	559
27	560
27.1	560
27.1.1 x64	560
27.1.2 x86	561
27.2	568
27.3	569
27.3.1 x64	569
27.3.2 x86	571
27.4 :x64 y SIMD	572
27.5	572
27.6	573
28 Detalles específicos de ARM	574

28.1 (#) Signo numeral (#) antes del número	574
28.2 Modos de direccionamiento	574
28.3 Cargar una constante en un registro	575
28.3.1 32 bits ARM	575
28.3.2 ARM64	576
28.4 en ARM64Relocaciones en ARM64	577
29 Detalles específicos de MIPS	580
29.1 Carga de constantes en un registro	580
29.2 Lecturas adicionales sobre MIPS	581
II	582
30	584
31 Endianess	586
31.1 Big-endian	586
31.2 Little-endian	586
31.3 Ejemplo	587
31.4 Bi-endian	587
31.5	587
32	588
33 CPU	589
33.1	589
33.2	589
34	590
34.1 ?	590
III	591
35	592
35.1	592
35.1.1 Con optimización MSVC 2012 x86	593
35.1.2 Con optimización MSVC 2012 x64	594
35.2	595
36	598
36.1 #1	598
36.2 #2	602
36.3	606

38.1 calc_network_address()	614
38.2 form_IP()	614
38.3 print_as_IP()	616
38.4 form_netmask() y set_bit()	618
38.5	619

39.1	620
39.2	621
39.3 Intel C++ 2011	623

40 Duff's device**628****41 División por 9****631**

41.1 x86	631
41.2 ARM	632
41.2.1 Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)	633
41.2.2 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)	633
41.2.3 Sin optimización Xcode 4.6.3 (LLVM) y Keil 6/2013	633
41.3 MIPS	634
41.4	634
41.4.1	635
41.5	636
41.5.1 #1	636
41.5.2 #2	637
41.6 Ejercicio #1	638

42 (atoi())**639**

42.1	639
42.1.1 Con optimización MSVC 2013 x64	639
42.1.2 Con optimización GCC 4.9.1 x64	640
42.1.3 Con optimización Keil 6/2013 (Modo ARM)	641
42.1.4 Con optimización Keil 6/2013 (Modo Thumb)	642
42.1.5 Con optimización GCC 4.9.1 ARM64	642
42.2	643
42.2.1 Con optimización GCC 4.9.1 x64	644
42.2.2 Con optimización Keil 6/2013 (Modo ARM)	646
42.3 Ejercicio	647

43**648**

43.1	649
------	-----

	ÍNDICE GENERAL
43.1.1 <code>strcmp()</code>	649
43.1.2 <code>strlen()</code>	652
43.1.3 <code>strcpy()</code>	652
43.1.4 <code>memset()</code>	653
43.1.5 <code>memcpy()</code>	654
43.1.6 <code>memcmp()</code>	658
43.1.7	659
44 C99 <code>restrict</code>	660
45	663
45.1 Con optimización GCC 4.9.1 x64	663
45.2 Con optimización GCC 4.9 ARM64	664
46	665
46.1	665
46.1.1	666
46.1.2	667
46.2	669
47	671
47.1 x64: Con optimización MSVC 2013	672
47.2 x64: Sin optimización GCC 4.9.1	674
47.3 x64: Con optimización GCC 4.9.1	675
47.4 ARM64: Sin optimización GCC (Linaro) 4.9	676
47.5 ARM64: Con optimización GCC (Linaro) 4.9	678
47.6 ARM: Con optimización Keil 6/2013 (Modo ARM)	679
47.7 ARM: Con optimización Keil 6/2013 (Modo Thumb)	680
47.8 MIPS	681
48	683
48.1 x64	683
48.1.1	683
48.1.2	684
48.2 ARM	685
48.2.1 GCC para ARM64	686
48.3	687
49	688
49.1 (x86)	688
49.2 ?	689
50	696
50.1	696
50.2	697

50.2.1	697
50.2.2	697
50.2.3	698
50.2.4	698
50.2.5	699
50.3	699
50.4	699
50.5 Ejercicios	699
50.5.1 Ejercicio #1	699
 51 C++		 701
51.1	701
51.1.1	701
51.1.2	708
51.1.3	712
51.1.4	714
51.1.5	718
51.2 ostream	721
51.3 References	723
51.4 STL	723
51.4.1 std::string	723
51.4.2 std::list	732
51.4.3 std::vector	746
51.4.4 std::map y std::set	756
 52		 772
 53 Windows 16-bit		 776
53.1 Ejemplo#1	776
53.2 Ejemplo #2	777
53.3 Ejemplo #3	778
53.4 Ejemplo #4	779
53.5 Ejemplo #5	783
53.6 Ejemplo #6	787
53.6.1	789
 IV Java		 792
 54 Java		 793
54.1	793
54.2	793
54.3	798
54.4	801

54.5	802
54.6 beep()	803
54.7	804
54.8	805
54.9	808
54.10	809
54.11	811
54.12switch()	813
54.13	814
54.13.1	814
54.13.2	816
54.13.3	816
54.13.4	817
54.13.5	819
54.13.6	822
54.13.7	823
54.13.8	824
54.14	824
54.14.1	824
54.14.2	825
54.15	827
54.16	831
54.17	833
54.17.1	833
54.17.2	836
54.18	839
V	840
55	842
55.1 Microsoft Visual C++	842
55.1.1 Name mangling	842
55.2 GCC	842
55.2.1 Name mangling	842
55.2.2 Cygwin	843
55.2.3 MinGW	843
55.3 Intel FORTRAN	843
55.4 Watcom, OpenWatcom	843
55.4.1 Name mangling	843
55.5 Borland	843
55.5.1 Delphi	844
55.6	846

56 (win32)	847
56.1 Windows API	847
56.2 tracer:	848
57	850
57.1	850
57.1.1 C/C++	850
57.1.2 Borland Delphi	851
57.1.3 Unicode	851
57.1.4 Base64	855
57.2	855
57.3	855
58	856
59	857
59.1 Magic numbers	858
59.1.1 DHCP	858
59.2	858
60	859
61	862
61.1	862
61.2	862
62	864
63	865
63.1	865
63.2 C++	865
63.3	865
63.4	866
63.4.1	867
63.4.2	867
VI	868
64	869
64.1 cdecl	869
64.2 stdcall	869
64.2.1	870
64.3 fastcall	870
64.3.1 GCC regparm	871

64.3.2 Watcom/OpenWatcom	871
64.4 thiscall	872
64.5 x86-64	872
64.5.1 Windows x64	872
64.5.2 Linux x64	875
64.6	875
64.7	875
64.8	876
65 Thread Local Storage	879
65.1	879
65.1.1 Win32	879
65.1.2 Linux	884
66	886
66.1 Linux	886
66.2 Windows	887
67 Linux	888
67.1 Código independiente de la posición	888
67.1.1 Windows	891
67.2 <i>LD_PRELOAD</i> en Linux	891
68 Windows NT	895
68.1 CRT (win32)	895
68.2 Win32 PE	899
68.2.1	900
68.2.2	901
68.2.3 Subsystem	901
68.2.4	902
68.2.5	902
68.2.6	903
68.2.7	904
68.2.8	906
68.2.9 .NET	906
68.2.10TLS	906
68.2.11	907
68.2.12Further reading	907
68.3 Windows SEH	907
68.3.1	907
68.3.2	914
68.3.3 Windows x64	933
68.3.4 SEH	938
68.4 Windows NT:	938

VII Herramientas	941
69 Desensamblador	942
69.1 IDA	942
70 Depurador	943
70.1 OllyDbg	943
70.2 GDB	943
70.3 tracer	943
71 Rastreo de llamadas al sistema	945
71.0.1 strace / dtruss	945
72 Descompiladores	946
73 Otras herramientas	947
VIII	948
74 (Windows Vista)	950
74.1	952
75	954
76 Buscaminas (Windows XP)	958
76.1 Ejercicios	964
77	965
77.1	965
77.2	970
78	976
78.1 #1: MacOS Classic y PowerPC	976
78.2 #2: SCO OpenServer	986
78.2.1	997
78.3 #3: MS-DOS	1001
79	1009
80 SAP	1050
80.1	1050
80.2	1065
81 Oracle RDBMS	1071
81.1	1071

81.2 Oracle RDBMS	1082
81.3 Oracle RDBMS	1084
82	1089
82.1 EICAR	1089
83	1091
83.1 10 PRINT CHR\$(205.5+RND(1)); : GOTO 10	1091
83.1.1	1092
83.1.2	1092
83.1.3	1093
83.1.4 Conclusión	1094
83.2	1095
83.2.1	1096
83.2.2	1102
83.2.3	1104
IX Ejemplos de ingeniería inversa de formatos de archivo propietarios	1107
84	1108
84.1 Norton Guide:	1109
84.1.1	1110
84.2	1112
84.2.1 Ejercicio	1115
85	1116
86 Oracle RDBMS:	1123
87 Oracle RDBMS: Archivos .MSB	1134
87.1	1139
X	1140
88 npad	1141
89	1144
89.1	1144
89.2 x86	1144
90 Compiler intrinsic	1145

91	1146
92 OpenMP	1148
92.1 MSVC	1150
92.2 GCC	1153
93 Itanium	1156
94	1160
95	1161
95.1 Profile-guided optimization	1161
XI	1163
96	1164
96.1 Windows	1164
96.2 C/C++	1164
96.3 x86 / x86-64	1164
96.4 ARM	1164
96.5	1164
97	1165
97.1 Windows	1165
98	1166
XII Ejercicios	1167
99 1	1170
99.1 Ejercicio 1.4	1170
100 2	1171
100.1 Ejercicio 2.1	1171
100.1.1 Con optimización MSVC 2010 x86	1171
100.1.2 Con optimización MSVC 2012 x64	1172
100.2 Ejercicio 2.4	1173
100.2.1 Con optimización MSVC 2010	1173
100.2.2 GCC 4.4.1	1174
100.2.3 Con optimización Keil (Modo ARM)	1176
100.2.4 Con optimización Keil (Modo Thumb)	1176
100.2.5 Con optimización GCC 4.9.1 (ARM64)	1177
100.2.6 Con optimización GCC 4.4.5 (MIPS)	1178

100.3 Ejercicio 2.6	1180
100.3.1Con optimización MSVC 2010	1180
100.3.2Con optimización Keil (Modo ARM)	1181
100.3.3Con optimización Keil (Modo Thumb)	1182
100.3.4Con optimización GCC 4.9.1 (ARM64)	1183
100.3.5Con optimización GCC 4.4.5 (MIPS)	1184
100.4 Ejercicio 2.13	1184
100.4.1Con optimización MSVC 2012	1184
100.4.2Keil (Modo ARM)	1185
100.4.3Keil (Modo Thumb)	1185
100.4.4Con optimización GCC 4.9.1 (ARM64)	1186
100.4.5Con optimización GCC 4.4.5 (MIPS)	1186
100.5 Ejercicio 2.14	1186
100.5.1MSVC 2012	1186
100.5.2Keil (Modo ARM)	1188
100.5.3GCC 4.6.3 for Raspberry Pi (Modo ARM)	1189
100.5.4Con optimización GCC 4.9.1 (ARM64)	1190
100.5.5Con optimización GCC 4.4.5 (MIPS)	1191
100.6 Ejercicio 2.15	1193
100.6.1Con optimización MSVC 2012 x64	1193
100.6.2Con optimización GCC 4.4.6 x64	1197
100.6.3Con optimización GCC 4.8.1 x86	1198
100.6.4Keil (Modo ARM):	1200
100.6.5Con optimización GCC 4.9.1 (ARM64)	1201
100.6.6Con optimización GCC 4.4.5 (MIPS)	1202
100.7 Ejercicio 2.16	1204
100.7.1 Con optimización MSVC 2012 x64	1204
100.7.2 Con optimización Keil (Modo ARM)	1204
100.7.3 Con optimización Keil (Modo Thumb)	1205
100.7.4 Sin optimización GCC 4.9.1 (ARM64)	1205
100.7.5 Con optimización GCC 4.9.1 (ARM64)	1206
100.7.6 Sin optimización GCC 4.4.5 (MIPS)	1210
100.8 Ejercicio 2.17	1212
100.9 Ejercicio 2.18	1212
100.10 Ejercicio 2.19	1212
100.11 Ejercicio 2.20	1213
101	1214
101.1 Ejercicio 3.2	1214
101.2 Ejercicio 3.3	1214
101.3 Ejercicio 3.4	1214
101.4 Ejercicio 3.5	1215
101.5 Ejercicio 3.6	1215
101.6 Ejercicio 3.8	1215

1219**1221****A x86****1221**

A.1	1221
A.2	1221
A.2.1 RAX/EAX/AX/AL	1222
A.2.2 RBX/EBX/BX/BL	1222
A.2.3 RCX/ECX/CX/CL	1222
A.2.4 RDX/EDX/DX/DL	1222
A.2.5 RSI/ESI/SI/SIL	1222
A.2.6 RDI/EDI/DI/DIL	1223
A.2.7 R8/R8D/R8W/R8L	1223
A.2.8 R9/R9D/R9W/R9L	1223
A.2.9 R10/R10D/R10W/R10L	1223
A.2.10 R11/R11D/R11W/R11L	1223
A.2.11 R12/R12D/R12W/R12L	1224
A.2.12 R13/R13D/R13W/R13L	1224
A.2.13 R14/R14D/R14W/R14L	1224
A.2.14 R15/R15D/R15W/R15L	1224
A.2.15 RSP/ESP/SP/SPL	1224
A.2.16 RBP/EBP/BP/BPL	1225
A.2.17 RIP/EIP/IP	1225
A.2.18 CS/DS/ES/SS/FS/GS	1225
A.2.19	1225
A.3 FPU registros	1226
A.3.1	1227
A.3.2	1227
A.3.3 Tag Word	1228
A.4 SIMD registros	1228
A.4.1 MMX registros	1228
A.4.2 SSE y AVX registros	1228
A.5	1228
A.5.1 DR6	1229
A.5.2 DR7	1229
A.6	1230
A.6.1	1231

A.6.2	1231
A.6.3	1236
A.6.4	1242
A.6.5	1244
B ARM		1247
B.1	1247
B.2	1247
B.3 32- ARM (AArch32)	1248
B.3.1	1248
B.3.2 Current Program Status Register (CPSR)	1248
B.3.3	1248
B.4 64- ARM (AArch64)	1249
B.4.1	1249
B.5	1249
B.5.1	1250
C MIPS		1251
C.1 Registros	1251
C.1.1 GPR⁴	1251
C.1.2	1252
C.2 Instrucciones	1252
C.2.1	1252
D		1253
E		1254
F Cheatsheets		1255
F.1 IDA	1255
F.2 OllyDbg	1256
F.3 MSVC	1257
F.4 GCC	1257
F.5 GDB	1257
G		1259
G.1	1259
G.1.1	1259
G.1.2	1259
G.1.3	1260
G.1.4	1261
G.1.5 Ejercicio #1	1261
G.1.6	1261

⁴General Purpose Registers

G.1.7	Ejercicio #3	1261
G.1.8	Ejercicio #4	1261
G.1.9	1262
G.1.10	1262
G.1.11	1262
G.1.12	1263
G.1.13	1264
G.1.14	1266
G.1.15	1267
G.1.16	1268
G.2	1	1268
G.2.1	Ejercicio 1.1	1268
G.2.2	Ejercicio 1.4	1268
G.3	2	1268
G.3.1	Ejercicio 2.1	1268
G.3.2	Ejercicio 2.4	1268
G.3.3	Ejercicio 2.6	1269
G.3.4	Ejercicio 2.13	1269
G.3.5	Ejercicio 2.14	1270
G.3.6	Ejercicio 2.15	1270
G.3.7	Ejercicio 2.16	1270
G.3.8	Ejercicio 2.17	1270
G.3.9	Ejercicio 2.18	1270
G.3.10	Ejercicio 2.19	1270
G.3.11	Ejercicio 2.20	1271
G.4	3	1271
G.4.1	Ejercicio 3.2	1271
G.4.2	Ejercicio 3.3	1271
G.4.3	Ejercicio 3.4	1271
G.4.4	Ejercicio 3.5	1271
G.4.5	Ejercicio 3.6	1271
G.4.6	Ejercicio 3.8	1271
G.5	1272
G.5.1	«Buscaminas (Windows XP)»	1272

Lista de acrónimos usados	1274
Glosario	1279
Índice alfabético	1281
Bibliografía	1290

Prefacio

Temas tratados en profundidad

x86/x64, ARM/ARM64, MIPS, Java/JVM.

Temas mencionados

Oracle RDBMS ([81 on page 1071](#)), Itanium ([93 on page 1156](#)), ([78 on page 976](#)), LD_PRELOAD ([67.2 on page 891](#)), , ELF⁵, ([68.2 on page 899](#)), x86-64 ([26.1 on page 550](#)), ([68.4 on page 938](#)), ([66 on page 886](#)), TLS⁶, (PIC⁷) ([67.1 on page 888](#)), profile-guided optimization ([95.1 on page 1161](#)), C++ STL ([51.4 on page 723](#)), OpenMP ([92 on page 1148](#)), SEH ([68.3 on page 907](#)).

Sobre el autor

[dennis\(a\)yurichev.com](mailto:dennis(a)yurichev.com), dennis.yurichev.



Ingeniería inversa para principiantes

- «It's very well done .. and for free .. amazing.»⁸ Daniel Bilar, Siege Technologies, LLC.
- «... excellent and free»⁹ Pete Finnigan, Oracle RDBMS.
- «... book is interesting, great job!» Michael Sikorski, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*.

⁵ Formato de archivo ejecutable usado ampliamente en sistemas *NIX, incluido Linux

⁶ Thread Local Storage

⁷ Position Independent Code: [67.1 on page 888](#)

⁸twitter.com/daniel_bilar/status/436578617221742593

⁹twitter.com/petefinnigan/status/400551705797869568

- «... my compliments for the very nice tutorial!» Herbert Bos, Vrije Universiteit Amsterdam, *Modern Operating Systems (4th Edition)*.
- «... It is amazing and unbelievable.» Luis Rocha, CISSP / ISSAP, Technical Manager, Network & Information Security at Verizon Business.
- «Thanks for the great work and your book.» Joris van de Vis, SAP Netweaver & Security.
- «... reasonable intro to some of the techniques.»¹⁰ Mike Stay, Federal Law Enforcement Training Center, Georgia, US.
- «I love this book! I have several students reading it at the moment, plan to use it in graduate course.»¹¹, Research Assistant Professor Dartmouth College
- «Dennis @Yurichev has published an impressive (and free!) book on reverse engineering»¹² Tanel Poder, .
- «This book is some kind of Wikipedia to beginners...» Archer, Chinese Translator, IT Security Researcher.

Agradecimientos

: Andrey «herm1t» Baranovich, .

: , Shell Rocket, Zhu Ruijin, Changmin Heo.

: , Arnaud Patard (rtp #debian-arm IRC), .

: Archer.

: Byungho Min.

: , , , “Logxen” , Yuan Jochen Kang, Mal Malakov, Lewis Porter, Jarle Thorsen.

2 * Oleg Vygovsky (50+100 UAH), Daniel Bilar (\$50), James Truscott (\$4.5), Luis Rocha (\$63), Joris van de Vis (\$127), Richard S Shultz (\$20), Jang Minchang (\$20), Shade Atlas (5 AUD), Yao Xiao (\$10), Paweł Szczur (40 CHF), Justin Simms (\$20), Shawn the R0ck (\$27), Ki Chan Ahn (\$50), Triop AB (100 SEK), Ange Albertini (€10+50), Sergey Lukianov (300 RUR), Ludvig Gislason (200 SEK), Gérard Labadie (€40), Sergey Volchkov (10 AUD), Vankayala Vigneswararao (\$50), Philippe Teuwen (\$4), Martin Haeberli (\$10), Victor Cazacov (€5), Tobias Sturzenegger (10 CHF), Sonny Thai (\$15), Bayna AlZaabi (\$75), Redfive B.V. (€25), Joona Oskari Heikkilä (€5), Marshall Bishop (\$50), Nicolas Werner (€12), Jeremy Brown (\$100), Alexandre Borges (\$25), Vladimir

¹⁰[reddit](#)

¹¹twitter.com/sergeybratus/status/505590326560833536

¹²twitter.com/TanelPoder/status/524668104065159169

Dikovski (€50), Jiarui Hong (100.00 SEK), Jim Di (500 RUR), Tan Vincent (\$30), Sri Harsha Kandrkota (10 AUD), Pillay Harish (10 SGD), Timur Valiev (230 RUR), Carlos Garcia Prado (€10), Salikov Alexander (500 RUR), Oliver Whitehouse (30 GBP), Katy Moe (\$14), Maxim Dyakonov (\$3), Sebastian Aguilera (€20), Hans-Martin Münch (€15), Jarle Thorsen (100 NOK), Vitaly Osipov (\$100), Yuri Romanov (1000 RUR), Aliaksandr Autayeу (€10), Tudor Azoitei (\$40), Z0vsky (€10), Yu Dai (\$10).

mini-FAQ

Q: ¿Por qué debería aprenderse el lenguaje ensamblador hoy en día?

A: A menos que seas desarrollador de un [OS¹³](#), probablemente no necesites programar en ensamblador: los compiladores modernos optimizan mucho mejor que los humanos¹⁴. Además, las [CPU¹⁵](#) modernas son dispositivos muy complejos y saber ensamblador difícilmente ayuda a comprender su funcionamiento interno. Aun así, hay al menos dos áreas donde comprender bien el ensamblador resulta útil: 1) el análisis de seguridad y malware; 2) entender mejor tu código compilado mientras depuras. Por ello, este libro está pensado para quienes quieren entender el ensamblador más que programar en él, de ahí la cantidad de ejemplos de salidas de compiladores.

Q: Hice clic en un hipervínculo dentro de un PDF, ¿cómo vuelvo atrás?

A: En Adobe Acrobat Reader pulsa Alt+Flecha Izquierda.

Q: ¡Tu libro es enorme! ¿Hay algo más corto?

A: Hay una versión abreviada (lite) aquí: <http://beginners.re/#lite>.

Q: No estoy seguro de si debería aprender ingeniería inversa o no.

A: Quizá el tiempo medio para familiarizarse con la versión abreviada LITE sea de 1 a 2 meses.

Q: ¿Puedo imprimir este libro? ¿Usarlo para enseñar?

A: Por supuesto; por eso el libro está bajo licencia Creative Commons. También puedes compilar tu propia versión del libro; lee [aquí](#) para saber más.

Q: Quiero traducir tu libro a otro idioma.

A: Lee [mi nota para traductores](#).

Q: ¿Cómo conseguir trabajo como ingeniero inverso?

A: En reddit dedicado a RE¹⁶ aparecen hilos de contratación de vez en cuando

¹³Sistema Operativo

¹⁴Un texto muy bueno sobre este tema: [\[Fog13b\]](#)

¹⁵Central processing unit

¹⁶reddit.com/r/ReverseEngineering/

([2013 Q3](#), [2014](#)). Échales un ojo. En el subred «netsec» hay un hilo parecido: [2014 Q2](#).

Q: ¿Dónde estudiar en Ucrania?

A: NTUU «KPI»: «Аналіз програмного коду та бінарних вразливостей»; optativas.

Q: Tengo una pregunta...

A: Envíamela por correo ([dennis\(a\)yurichev.com](mailto:dennis(a)yurichev.com)).

. . . : [beginners.re](#).

Sobre la traducción al coreano

Byungho Min ([twitter/tais9](#)).

: [facebook/andydinka](#).

Parte I

Patrones de código

Cuando el autor de este libro comenzó a aprender C y, más tarde, C++, él solía escribir pequeños trozos de código, compilarlos, y luego ver los resultados en lenguaje assembly. Esto lo hizo muy fácil para él entender lo que estaba pasando en el código que había escrito.¹⁷ Él lo hizo tantas veces que la relación entre el código C/C++ y lo que el compilador producido se imprimió profundamente en su mente. Es fácil imaginar al instante un esbozo de la apariencia y función del código C. Quizás esta técnica podría ser útil para otra persona.

En ciertas partes, se han empleado aquí compiladores muy antiguas, con el fin de obtener lo mas corta (o simple) posible snippet.

Ejercicios

Cuando el autor de este libro estudió la lenguaje assembly, también con frecuencia compilaba pequeñas funciones en C, y reescribia gradualmente en assembly, tratando de hacer el código lo más pequeño posible. Probablemente no vale la pena hacer esto en escenarios reales actualmente, porque es difícil competir con los compiladores modernos en términos de eficiencia. Es, sin embargo, una muy buena manera de obtener una mejor comprensión de la assembly

Siéntase libre, por lo tanto, para tomar cualquier código de este libro y tratar de hacerlo más pequeño. Sin embargo, no se olvide de probar lo que has escrito.

Niveles de optimización y la información de depuración

El código fuente puede ser compilado por diferentes compiladores com varios niveles de optimización. Un compilador típico tiene alredor de tres de esos niveles, donde el nivel cero significa desactivar la optimización. La optimización también puede dirigirse hacia el tamaño del código o la velocidad de código.

Un compilador sin optimización es más rápido y produce código más inteligible (aunque más grande), mientras un compilador con optimización es más lento y trata de producir un código que corre más rápido (pero no necesariamente más compacto).

Además de los niveles y dirección de la optimización, el compilador puede incluir informaciones de depuración en el archivo resultante, produciendo así código para fácil depuración.

¹⁷De hecho, todavía lo hace cuando no puede entender lo que hace una determinada pieza de código.

Una de los características importantes del código de ‘debug’ es que puede contener enlaces entre cada línea del código fuente y las direcciones de código de máquina respectivos. Compiladores con optimización, por otro lado, tienden a producir una salida donde líneas enteras de código fuente pueden ser optimizados al punto de ser eliminados y por consiguiente no estar presentes en el código de máquina resultante.

Ingenieros Inversos pueden encontrar ambas versiones, simplemente porque algunos desarrolladores activan los flags de optimización del compilador, y otros no activan. Debido a esto, vamos a tratar de trabajar con ejemplos de ambas versiones de debug y release del código resaltado en este libro, cuando sea posible.

Capítulo 1

Una breve introducción a la CPU

La **CPU** es el dispositivo que ejecuta el código de máquina que constituye un programa.

Un breve glosario:

Instrucción : Una primitiva **CPU** comando. Los ejemplos más simples incluyen: mover datos entre registros, trabajar con la memoria, operaciones aritméticas primitivas. Como regla general, cada **CPU** tiene su propio conjunto de instrucciones (**ISA**).

: Código que la **CPU** procesa directamente. Cada instrucción generalmente se codifica por varios bytes.

Lenguaje assembly : Código mnemónico y algunas extensiones como macros que destinados a hacer la vida del programador más fácil.

Registros de la CPU : Cada **CPU** tiene un conjunto fijo de registros de propósito general (**GPR**). ≈ 8 en x86, ≈ 16 en x86-64, ≈ 16 en ARM. La forma más fácil de entender un registro es pensar en ello como una variable temporal sin tipo. Imagine si estuviera trabajando con una **PL**¹ de alto nivel y sólo podría utilizar ocho variables de 32-bit (o de 64-bit). Sin embargo mucho se puede hacer usando sólo estos!

Uno podría preguntarse por qué es necesario que haya diferencia entre el código de la máquina y una lenguaje de programación de alto nivel. La respuesta está en el hecho de que los seres humanos y CPUs no son iguales. Es mucho más fácil para los humanos utilizar un **PL** de alto nivel como C/C++, Java, Python, etc., pero es más fácil para una **CPU** utilizar un nivel mucho más bajo de abstracción. Tal vez sería posible inventar una **CPU** que podría ejecutar código de **PL** de alto nivel, pero sería

¹Lenguaje de programación

muchas veces más compleja que las CPUs que conocemos hoy. En una manera similar, es muy incómodo para los seres humanos escribir en lenguaje assembly, debido a que es tan bajo nivel y difícil escribir sin hacer una gran cantidad de errores molestos. El programa que convierte el código de PL de alto nivel en assembly se llama *compiler*.

1.1 Algunas palabras sobre diferentes ISAs

El ISA x86 siempre ha tenido opcodes de tamaño variable, de modo que cuando llegó la era de 64-bit, las extensiones x64 no impactan el ISA de manera muy significativa. De hecho, el ISA x86 aún contiene una gran cantidad de instrucciones que primero aparecieron en CPU 8086 16-bit, pero aún se encuentran en las CPUs de hoy.

ARM es una CPU RISC² diseñado con la idea de opcodes con tamaño constante, que tenía algunas ventajas en el pasado. En el principio, todas las instrucciones ARM fueron codificados en 4 bytes³. Esto actualmente se conoce como «ARM mode».

Entonces se llegó a la conclusión que no era tan económico como se imaginó al principio. En realidad, la mayoría de las instrucciones de CPU utilizados⁴ en aplicaciones del mundo real pueden ser codificados utilizando menos información. Por lo tanto añadieron otra ISA, llamado Thumb, donde cada instrucción fue codificada en sólo 2 bytes. Esto se conoce como «Thumb mode». No obstante, no todas las instrucciones ARM pueden ser codificadas en apenas 2 bytes, entonces el conjunto de instrucciones Thumb es algo limitada. Es importante destacar que el código compilado para el modo ARM y para el modo Thumb pueden, por supuesto, coexistir dentro de un solo programa.

Los creadores de ARM concluyeron que se podría extender el Thumb, dando origen al Thumb-2, que apareció en el ARMv7. Thumb-2 sigue utilizando instrucciones de 2 bytes, pero tiene algunas nuevas instrucciones que tienen el tamaño de 4 bytes. Hay una idea errónea de que Thumb-2 es una mezcla de ARM y Thumb. Esto es incorrecto. Más bien, se extendió Thumb-2 para apoyar plenamente todas las características de procesador por lo que podría competir con el modo ARM un objetivo que se logró con claridad, ya que la mayoría de aplicaciones para iPod/iPhone/iPad son compilados para el conjunto de instrucciones del Thumb-2 (la verdad es, en gran parte debido al hecho de que Xcode hace esto por defec-

²Reduced instruction set computing

³Dicho sea de paso, las instrucciones de longitud fija son muy útiles porque se puede calcular la dirección de instrucción siguiente (o anterior) sin esfuerzo. Esta característica se discutirá en la sección de el operador switch() (13.2.2 on page 203).

⁴Son estos MOV/PUSH/CALL/Jcc

to). Más tarde, el ARM 64-bit salió. Este ISA tiene opcodes de 4 bytes, y descarta la necesidad de cualquier modo Thumb adicional. Pero, los requisitos de 64-bit afectaron la ISA, resultando en ahora tenermos tres conjuntos de instrucciones ARM: ARM mode, Thumb mode (incluyendo Thumb-2) y ARM64. Estos ISAs se intersectan parcialmente, pero puede ser más bien decir que son ISAs diferentes, en lugar de variaciones de lo mismo. Por lo tanto, nos gustaría intentar añadir fragmentos de código de los tres ISAs del ARM en este libro.

Hay, por cierto, muchos otros RISC ISAs con opcodes de tamaño fijo de 32-bit, tales como MIPS, PowerPC y Alpha AXP.

Capítulo 2

La función más simple

La función más simple posible es, probablemente, aquella que simplemente devuelve un valor constante:

: Aquí está:

Listing 2.1: Código C/C++

```
int f()
{
    return 123;
}
```

¡Compilémosla!

2.1 x86

: Esto es lo que producen los compiladores GCC y MSVC con optimización en la plataforma x86:

Listing 2.2: Con optimización GCC/MSVC (salida de assembly)

```
f:
    mov     eax, 123
    ret
```

Solo hay dos instrucciones: la primera coloca el valor 123 en el registro EAX, que por convención se usa para almacenar el valor de retorno, y la segunda es RET, que devuelve la ejecución a la [caller](#). La función llamadora tomará el resultado del registro EAX.

2.2 ARM

Hay algunas diferencias en la plataforma ARM:

Listing 2.3: Con optimización Keil 6/2013 (Modo ARM) ASM Output

```
f PROC
    MOV      r0,#0x7b ; 123
    BX       lr
ENDP
```

ARM utiliza el registro R0 para devolver los resultados de las funciones, por lo que 123 se copia en R0.

En ARM la dirección de retorno no se guarda en la pila local, sino en el registro de enlace ([LR¹](#)). Por ello, la instrucción BX LR salta a esa dirección, lo que equivale a devolver la ejecución a la [caller](#).

Vale la pena señalar que MOV es un nombre equívoco para la instrucción tanto en x86 como en las [ISA ARM](#). En realidad, los datos no se *mueven*, sino que se *copian*.

2.3 MIPS

Hay dos convenciones para nombrar registros en MIPS. por número (de \$0 a \$31) o por seudónimo (\$V0, \$A0, Spanish text placeholder). La salida de ensamblador de GCC a continuación muestra los registros por número:

Listing 2.4: Con optimización GCC 4.4.5 (salida de assembly)

```
j      $31
li     $2,123          # 0x7b
```

...: ... mientras que [IDA²](#) los muestra por sus seudónimos:

Listing 2.5: Con optimización GCC 4.4.5 (IDA)

```
jr    $ra
li    $v0, 0x7B
```

Así que el registro \$2 (o \$V0) se usa para almacenar el valor de retorno de la función. LI “Load Immediate” . significa “Load Immediate” y es el equivalente a MOV en MIPS.

¹Link Register

²Desensamblador y depurador interactivo desarrollado por [Hex-Rays](#)

La otra instrucción es la de salto (`J` o `JR`), que devuelve la ejecución a la [caller](#) saltando a la dirección contenida en el registro `$31` (o `$RA`). Este registro es análogo al [LR](#) en ARM.

“branch delay slot”. Quizá te preguntes por qué la instrucción de carga (`LI`) y la instrucción de salto (`J` o `JR`) están intercambiadas. Esto se debe a una característica de las arquitecturas [RISC](#) llamada “branch delay slot”. En realidad, no necesitamos entrar en estos detalles. Solo hay que recordar que en MIPS, la instrucción que sigue a una instrucción de salto se ejecuta *antes* que la propia instrucción de salto/ramificación. Como consecuencia, las instrucciones de salto siempre intercambian su lugar con la instrucción que debe ejecutarse previamente.

2.3.1 Una nota sobre los nombres de instrucciones/registros en MIPS

En el mundo MIPS, los nombres de registros e instrucciones tradicionalmente se escriben en minúsculas. Sin embargo, para mantener la coherencia, usaremos mayúsculas, ya que es la convención seguida por las demás [ISA](#) presentadas en este libro.

Capítulo 3

Hello, world!

Continuemos usando el famoso ejemplo del libro “The C programming Language”[Ker88]:

```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
```

3.1 x86

3.1.1 MSVC

MSVC 2010:

```
c1 1.cpp /Fa1.asm
```

Listing 3.1: MSVC 2010

```
CONST SEGMENT
$SG3830 DB      'hello, world', 0AH, 00H
CONST ENDS
PUBLIC _main
EXTRN _printf:PROC
; Function compile flags: /Odtp
_TEXT SEGMENT
```

```

_main PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG3830
    call    _printf
    add    esp, 4
    xor    eax, eax
    pop    ebp
    ret    0
_main ENDP
_TEXT ENDS

```

3.1.3 on the next page.

:

```

#include <stdio.h>

const char $SG3830[]="hello, world\n";

int main()
{
    printf($SG3830);
    return 0;
}

```

. : 57.1.1 on page 850.

:main().¹.

:CALL _printf.

ADD ESP, 4 ADD ESP, 4

:

Listing 3.2: Oracle RDBMS 10.2 Linux (app.o)

.text:0800029A	push	ebx
.text:0800029B	call	qksfroChild
.text:080002A0	pop	ecx

XOR EAX, EAX.² MOV EAX, 0

SUB EAX, EAX,

¹ (4 on page 32).

² wikipedia

3.1.2 GCC

:gcc 1.c -o 1 [IDA](#),³.

Listing 3.3: IDA

```
main          proc near
var_10        = dword ptr -10h
              push    ebp
              mov     ebp, esp
              and     esp, 0FFFFFFF0h
              sub     esp, 10h
              mov     eax, offset aHelloWorld ; "hello, world\n"
              mov     [esp+10h+var_10], eax
              call    _printf
              mov     eax, 0
              leave
              retn
main          endp
```

SUB ESP, 10h

MOV EBP, ESP / AND ESP, ...).

3.1.3 GCC: Sintaxis AT&T

Listing 3.4: GCC 4.7.3

```
gcc -S 1_1.c
```

Listing 3.5: GCC 4.7.3

```
.file   "1_1.c"
.section .rodata
.LC0:
.string "hello, world\n"
.text
.globl main
.type   main, @function
main:
.LFB0:
.cfi_startproc
pushl  %ebp
.cfi_def_cfa_offset 8
```

³-S -masm=intel.

```

.cfi_offset 5, -8
movl    %esp, %ebp
.cfi_def_cfa_register 5
andl    $-16, %esp
subl    $16, %esp
movl    $.LC0, (%esp)
call    printf
movl    $0, %eax
leave
.cfi_restore 5
.cfi_def_cfa 4, 4
ret
.cfi_endproc
.LFE0:
.size   main, .-main
.ident  "GCC: (Ubuntu/Linaro 4.7.3-1ubuntu1) 4.7.3"
.section .note.GNU-stack,"",@progbits

```

4:

Listing 3.6: GCC 4.7.3

```

.LC0:
.string "hello, world\n"
main:
pushl  %ebp
movl    %esp, %ebp
andl    $-16, %esp
subl    $16, %esp
movl    $.LC0, (%esp)
call    printf
movl    $0, %eax
leave
ret

```

- : (\rightarrow)⁵.
- AT&T:
- AT&T:
 - q – quad (64)
 - l – long (32)
 - w – word (16)

⁴: *-fno-asynchronous-unwind-tables*

⁵

- b – byte (8)

```
:0FFFFFFF0h $-16.:16 0x10 . -0x10 0xFFFFFFFF0 ().
```

```
.MOV.
```

3.2 x86-64

3.2.1 MSVCx86-64

:

Listing 3.7: MSVC 2012 x64

```
$SG2989 DB      'hello, world', 0AH, 00H

main    PROC
        sub     rsp, 40
        lea     rcx, OFFSET FLAT:$SG2989
        call    printf
        xor     eax, eax
        add     rsp, 40
        ret     0
main    ENDP
```

. (fastcall: [64.3 on page 870](#)).. RCX, RDX, R8, R9..

..

RAX/EAX/AX/AL x86-64:

Spanish text placeholder		
RAX ^{x64}		
EAX		
AX		
AH	AL	

. «shadow space», : [8.2.1 on page 118](#).

3.2.2 GCCx86-64

:

Listing 3.8: GCC 4.4.6 x64

```
.string "hello, world\n"
main:
```

```

sub    rsp, 8
mov    edi, OFFSET FLAT:.LC0 ; "hello, world\n"
xor    eax, eax ;
call   printf
xor    eax, eax
add    rsp, 8
ret

```

[Mit13]. RDI, RSI, RDX, RCX, R8, R9.

. , RDI?

[Int13]. MOV EAX, 011223344h .

⁶:

Listing 3.9: GCC 4.4.6 x64

```

.text:00000000004004D0          main  proc near
.text:00000000004004D0 48 83 EC 08      sub    rsp, 8
.text:00000000004004D4 BF E8 05 40 00  mov    edi, offset ↴
    ↳ format ; "hello, world\n"
.text:00000000004004D9 31 C0          xor    eax, eax
.text:00000000004004DB E8 D8 FE FF FF  call   _printf
.text:00000000004004E0 31 C0          xor    eax, eax
.text:00000000004004E2 48 83 C4 08      add    rsp, 8
.text:00000000004004E6 C3            retn
.text:00000000004004E6               main  endp

```

.

. : «with variable arguments passes information about the number of vector registers used» [Mit13].

3.3 GCC

const (3.1.1 on page 11), ..

:

```
#include <stdio.h>

int f1()
{
    printf ("world\n");
}
```

⁶Options → Disassembly → Number of opcode bytes

```

int f2()
{
    printf ("hello world\n");
}

int main()
{
    f1();
    f2();
}

```

:

Listing 3.10: GCC 4.8.1 + IDA

```

f1          proc near
s           = dword ptr -1Ch
sub        esp, 1Ch
mov         [esp+1Ch+s], offset s ; "world\n"
call        _puts
add         esp, 1Ch
retn
f1          endp

f2          proc near
s           = dword ptr -1Ch
sub        esp, 1Ch
mov         [esp+1Ch+s], offset aHello ; "hello "
call        _puts
add         esp, 1Ch
retn
f2          endp

aHello      db 'hello '
s           db 'world',0xa,0

```

,
..
puts() «world» . puts()!

3.4 ARM

Para mis experimentos con procesadores ARM, se usaron varios compiladores:

- Popular en el entorno embebido Keil Release 6/2013.
- Apple Xcode 4.6.3 IDE (con LLVM-GCC 4.2 como compilador⁷).
- GCC 4.9 (Linaro) (para ARM64), disponible como ejecutables para win32 en <http://go.yurichev.com/17325>.

En todo este libro, salvo que se indique lo contrario, se usa ARM de 32 bits (incluidos los modos Thumb y Thumb-2). Cuando hablamos de ARM de 64 bits aquí, lo llamamos ARM64.

3.4.1 Sin optimización Keil 6/2013 (Modo ARM)

Para empezar, compilamos nuestro ejemplo en Keil:

```
armcc.exe --arm --c90 -O0 1.c
```

El compilador *armcc* produce listados en ensamblador con sintaxis Intel, pero incluye macros de alto nivel relacionados con ARM⁸, pero nos importa más ver las instrucciones tal cual, así que veamos el resultado compilado en [IDA](#).

Listing 3.11: Sin optimización Keil 6/2013 (Modo ARM) [IDA](#)

```
.text:00000000          main
.text:00000000 10 40 2D E9    STMFD   SP!, {R4,LR}
.text:00000004 1E 0E 8F E2    ADR     R0, aHelloWorld ; "hello, world"
.text:00000008 15 19 00 EB    BL      _2printf
.text:0000000C 00 00 A0 E3    MOV     R0, #0
.text:00000010 10 80 BD E8    LDMFD   SP!, {R4,PC}

.text:000001EC 68 65 6C 6C+aHelloWorld DCB "hello, world",0
; DATA XREF: main+4
```

En el ejemplo anterior se ve fácilmente que cada instrucción mide 4 bytes. Efectivamente, compilamos el código para modo ARM, no Thumb.

La primera instrucción, *STMFD SP!, {R4,LR}*⁹, funciona como una PUSH de x86, escribiendo los valores de dos registros (R4 y LR) en la pila. *PUSH {r4,lr}*

⁷Es así: Apple Xcode 4.6.3 usa GCC de código abierto como compilador de frontend y LLVM como generador de código

⁸p. ej., muestra instrucciones PUSH/POP, ausentes en el modo ARM

⁹*STMFD*¹⁰

de hecho muestra la instrucción. . Pero eso no es del todo preciso: PUSH solo está disponible en modo Thumb; para evitar confusiones, lo veremos en [IDA](#).

Esta instrucción primero [decrements](#) el [SP](#)¹¹ para que apunte a un lugar libre de la pila y después guarda los valores de R4 y [LR](#) en la dirección indicada por el [SP](#) modificado.

. Esta instrucción (como PUSH en modo Thumb) puede guardar varios registros a la vez, lo cual puede ser muy útil. Por cierto, esto no existe en x86. También puede notarse que STMFD es una generalización de PUSH (amplía sus capacidades), porque puede trabajar con cualquier registro y no solo con [SP](#). En otras palabras, STMFD puede usarse para almacenar un conjunto de registros en la dirección de memoria indicada.

La ADR R0, aHelloWorld hello, world instrucción suma o resta el valor de [PC](#)¹² al desplazamiento donde está la cadena. ¿Cómo se usa aquí [PC](#)? Se trata del llamado «código independiente de la posición». ¹³ Este código puede ejecutarse en una dirección no fija de memoria. En otras palabras, es direccionamiento relativo al [PC](#). El opcode de ADR codifica la diferencia entre la dirección de esta instrucción y el lugar donde está la cadena. Esta diferencia (desplazamiento) siempre será la misma, independientemente de la dirección a la que el [OS](#) cargue nuestro código. Por eso, lo único necesario es sumar la dirección de la instrucción actual (desde [PC](#)) para obtener la dirección absoluta de nuestra cadena en C.

BL __2printf¹⁴ . llama a la función printf(). Así es como funciona esta instrucción:

- guardar la dirección siguiente a BL (0xC) en [LR](#);
- luego transferir el control a printf() escribiendo su dirección en el registro [PC](#).

Cuando printf() termina debe saber adónde volver; por eso toda función devuelve el control a la dirección guardada en [LR](#).

¹⁵ .

.Por cierto, una dirección absoluta de 32 bits (o un desplazamiento) no puede codificarse en la instrucción de 32 bits BL, porque solo hay 24 bits disponibles. Como recordamos, todas las instrucciones en modo ARM miden 4 bytes (32 bits) y solo pueden situarse en direcciones múltiplos de 4; por ello, los 2 bits bajos de la dirección (siempre a 0) no se codifican. $current_PC \pm \approx 32M$. En resumen, tenemos 26 bits para codificar el desplazamiento; esto basta para $current_PC \pm \approx 32M$.

¹¹stack pointer. SP/ESP/RSP en x86/x64. SP en ARM.

¹²Program Counter. IP/EIP/RIP en x86/64. PC en ARM.

¹³Lee más sobre esto en la sección correspondiente ([67.1 on page 888](#))

¹⁴Branch with Link

¹⁵Lee más sobre esto en la siguiente sección ([5 on page 33](#))

A continuación, la `MOV R0, #016` simplemente escribe 0 en el registro R0. Nuestra función en C devuelve 0 y el valor de retorno debe ponerse en R0.

La última instrucción `LDMFD SP!, R4, PC17` es la inversa de `STMFD`. Carga desde la pila (o cualquier otra zona de memoria) los valores para guardarlos en R4 y PC, e **incrementa** el puntero de pila `SP`. Aquí actúa como un `POP`.

N.B. La primera instrucción `STMFD` guardó en la pila el par de registros R4 y `LR`, pero durante `LDMFD` se *restauran* R4 y `PC`.

Como ya sabemos, en `LR` suele guardarse la dirección a la que debe devolver el control cada función. La primera instrucción guarda ese valor en la pila, porque nuestra función `main()` usará ese registro al llamar a `printf()`. Al final de la función, ese valor puede escribirse directamente en `PC`, devolviendo el control al lugar desde el que fue llamada.

Como `main()` suele ser la función principal en C/C++, el control volverá al cargador del `OS`, o a algún punto del `CRT19`, o similar.

Todo esto permite omitir la instrucción `BX LR` al final de la función.

`DCB` es una directiva del lenguaje ensamblador que define un array de bytes o cadenas ASCII, similar a la directiva `DB` en ensamblador x86.

3.4.2 Sin optimización Keil 6/2013 (Modo Thumb)

Compilamos el mismo ejemplo en Keil en modo Thumb:

```
armcc.exe --thumb --c90 -00 1.c
```

Obtenemos (en `IDA`):

Listing 3.12: Sin optimización Keil 6/2013 (Modo Thumb) + [IDA](#)

```
.text:00000000          main
.text:00000000 10 B5      PUSH    {R4,LR}
.text:00000002 C0 A0      ADR     R0, aHelloWorld ; "hello,⤦
                           ↴ world"
.text:00000004 06 F0 2E F9      BL      __2printf
.text:00000008 00 20      MOVS   R0, #0
.text:0000000A 10 BD      POP    {R4,PC}

.text:00000304 68 65 6C 6C+aHelloWorld  DCB "hello, world",0      ↴
                           ↴ ; DATA XREF: main+2
```

¹⁶MOVE

¹⁷[LDMFD¹⁸](#)

¹⁹C runtime library : [68.1 on page 895](#)

Saltan a la vista los opcodes de 2 bytes (16 bits); como ya se señaló, es Thumb. Excepto la instrucción BL. .En realidad consta de dos instrucciones de 16 bits. .Esto es porque en un solo opcode de 16 bits hay muy poco espacio para especificar el desplazamiento hasta la función printf(). .Así que la primera instrucción de 16 bits carga los 10 bits altos del desplazamiento y la segunda los 11 bits bajos.. Como se mencionó, todas las instrucciones en modo Thumb tienen longitud de 2 bytes (o 16 bits). Por tanto, es imposible que una instrucción Thumb empiece en una dirección impar. . Dado lo anterior, el último bit de la dirección puede omitirse al codificar instrucciones. *current_PC* $\pm \approx 2M$.En resumen, en la instrucción Thumb BL se puede codificar la dirección *current_PC* $\pm \approx 2M$.

. El resto de instrucciones de la función (PUSH y POP) funcionan aquí casi igual que las STMFD/LDMFD descritas; solo que el registro SP no se indica explícitamente. ADR funciona igual que en el ejemplo anterior. MOVS .escribe 0 en el registro R0 para devolver cero.

3.4.3 Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Xcode 4.6.3 sin optimización genera demasiado código redundante, así que estudiaremos la salida optimizada, donde hay el menor número de instrucciones posible, activando el commutador del compilador -O3.

Listing 3.13: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```

__text:000028C4          _hello_world
__text:000028C4 80 40 2D E9    STMFD      SP!, {R7,LR}
__text:000028C8 86 06 01 E3    MOV         R0, #0x1686
__text:000028CC 0D 70 A0 E1    MOV         R7, SP
__text:000028D0 00 00 40 E3    MOVT        R0, #0
__text:000028D4 00 00 8F E0    ADD         R0, PC, R0
__text:000028D8 C3 05 00 EB    BL          _puts
__text:000028DC 00 00 A0 E3    MOV         R0, #0
__text:000028E0 80 80 BD E8    LDMFD      SP!, {R7,PC}

__cstring:00003F62 48 65 6C 6C+aHelloWorld_0  DCB "Hello world\r
\0"

```

Las instrucciones STMFD y LDMFD ya nos son familiares.

.La instrucción MOV simplemente escribe el número 0x1686 en R0: es el desplazamiento que apunta a la cadena «Hello world!».

El registro R7 (según el estándar de [App10]) es el frame pointer; lo veremos más abajo.

MOVT R0, #0 (MOVe Top) . escribe 0 en los 16 bits altos del registro. . El asunto es que la MOV genérica en modo ARM solo puede escribir en los 16 bits bajos del

registro. Recuerda que en modo ARM todos los opcodes están limitados a 32 bits. Claro que esto no afecta a los movimientos de datos entre registros. . Por eso existe la instrucción adicional MOVT para escribir en los bits altos (del 16 al 31 inclusive). . Su uso aquí es redundante, porque MOV R0, #0x1686 ya limpió la parte alta del registro. . Probablemente sea una limitación del compilador.

ADD R0, PC, R0 . suma PC a R0 para calcular la dirección absoluta de la cadena «Hello world!». Como ya sabemos, es «código independiente de la posición», así que esta corrección es esencial.

. La instrucción BL llama a puts() en lugar de printf().

El compilador sustituyó printf() por puts(). En efecto, printf() con un único argumento es casi equivalente a puts(). *Casi*, porque ambas funciones producen el mismo resultado solo si la cadena no contiene especificadores de formato de printf() que empiezan por %. Si los contiene, el efecto será diferente²⁰.

¿Por qué el compilador sustituyó printf() por puts()? Probablemente porque puts() es más rápida²¹. Seguramente porque puts() envía los caracteres a stdout sin comparar cada uno con el símbolo %.

A continuación, vemos la conocida MOV R0, #0 instrucción para establecer a 0 el valor devuelto en R0.

3.4.4 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

Xcode 4.6.3 :

Listing 3.14: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

__text:00002B6C          _hello_world
__text:00002B6C 80 B5      PUSH      {R7,LR}
__text:00002B6E 41 F2 D8 30    MOVW     R0, #0x13D8
__text:00002B72 6F 46      MOV       R7, SP
__text:00002B74 C0 F2 00 00    MOVT.W   R0, #0
__text:00002B78 78 44      ADD      R0, PC
__text:00002B7A 01 F0 38 EA    BLX     _puts
__text:00002B7E 00 20      MOVS    R0, #0
__text:00002B80 80 BD      POP     {R7,PC}

...
__cstring:00003E70 48 65 6C 6C 6F 20+aHelloWorld  DCB "Hello ↴
           ↴ world!",0xA,0

```

²⁰También hay que notar que puts() no requiere el símbolo de nueva línea '\n' al final de la cadena, por eso aquí no aparece.

²¹ciselant.de/projects/gcc_printf/gcc_printf.html

- ...
- :4-3-2-1;
 - :2-1;
 - :2-1-4-3.

0xFx.

```
MOVW R0, #0x13D8
MOVT.W R0, #0 MOVT Thumb-2.
BLX BL. :
```

<code>__symbolstub1:00003FEC _puts</code>	<code>; CODE XREF: ↴</code>
<code> ↴ _hello_world+E</code>	
<code>__symbolstub1:00003FEC 44 F0 9F E5</code>	<code>LDR PC, =__imp__puts</code>

(Windows PE .exe, ELF Mach-O) .

`_imp_puts ..`

“A piece of coding which provides an address;”, according to P. Z. Ingberman, who invented thunks in 1961 as a way of binding actual parameters to their formal definitions in Algol-60 procedure calls. If a procedure is called with an expression in the place of a formal parameter, the compiler generates a thunk which computes the expression and leaves the address of the result in some standard location.

...
 Microsoft and IBM have both defined, in their Intel-based systems, a “16-bit environment” (with bletcherous segment registers and 64K address limits) and a “32-bit environment” (with flat addressing and semi-real memory management). The two environments can both be running on the same computer and OS (thanks

to what is called, in the Microsoft world, WOW which stands for Windows On Windows). MS and IBM have both decided that the process of getting from 16- to 32-bit and vice versa is called a “thunk”; for Windows 95, there is even a tool THUNK.EXE called a “thunk compiler”.

([The Jargon File](#))

3.4.5 ARM64

GCC

Compilamos el ejemplo con GCC 4.8.1 en ARM64:

Listing 3.15: Sin optimización GCC 4.8.1 + objdump

```

1 0000000000400590 <main>:
2   400590:    a9bf7bfd      stp    x29, x30, [sp,#-16]!
3   400594:    910003fd      mov    x29, sp
4   400598:    90000000      adrp   x0, 400000 <_init-0x3b8>
5     ↴
6   40059c:    91192000      add    x0, x0, #0x648
7   4005a0:    97fffffa0    bl     400420 <puts@plt>
8   4005a4:    52800000      mov    w0, #0x0
9     ↴           // #0
10  4005a8:    a8c17bfd      ldp    x29, x30, [sp],#16
11  4005ac:    d65f03c0      ret
12 ...
13 Contents of section .rodata:
14   400640 01000200 00000000 48656c6c 6f210a00 .....Hello!..

```

En ARM64 no existen los modos Thumb ni Thumb-2, solo ARM, así que solo hay instrucciones de 32 bits. La cantidad de registros se duplica: [B.4.1 on page 1249](#). Los registros de 64 bits tienen el prefijo X-W-, mientras que sus partes de 32 bits – W-.

STP (*Store Pair*) guarda dos registros en la pila simultáneamente: X29 en X30. Por supuesto, esta instrucción puede guardar ese par en cualquier lugar de la memoria, pero aquí se especifica el registro SP, así que el par se guarda en la pila. Los registros en ARM64 son de 64 bits y cada uno ocupa 8 bytes, por lo que se necesitan 16 bytes para guardar dos registros.

El signo de exclamación tras el operando significa que primero se resta 16 de SP y solo después los valores del par de registros se escriben en la pila. A esto también

se le llama *pre-index*. Sobre la diferencia entre *post-index* y *pre-index* lee aquí: [28.2 on page 574](#).

Por lo tanto, en términos del más familiar x86, la primera instrucción es análoga a un par de PUSH X29 y PUSH X30. X29 , X30 [LR](#); ; por eso se guardan en el prólogo de la función y se restauran en el epílogo.

La segunda instrucción copia SP en X29 (o FP²²). Esto se hace para establecer el marco de pila de la función.

ADRP y ADD se usan para formar la dirección de la cadena «Hello!» en el registro X0, porque el primer argumento de la función se pasa por ese registro. (porque la longitud de la instrucción está limitada a 4 bytes; más sobre esto aquí: [28.3.1 on page 575](#)). Así que deben usarse varias instrucciones. La primera instrucción (ADRP) escribe en X0 la dirección de la página de 4 KB donde está la cadena, y la segunda (ADD) simplemente suma el resto a esa dirección. Más sobre esto en: [28.4 on page 577](#).

$0x400000 + 0x648 = 0x400648$, y vemos que en la sección de datos .rodata en esa dirección está nuestra cadena en C «Hello!».

Luego se llama a puts() usando la instrucción BL. Esto ya se comentó: [3.4.3 on page 21](#).

MOV La instrucción MOV escribe 0 en W0. W0 son los 32 bits bajos del registro de 64 bits X0:

Spanish text placeholder	Spanish text placeholder
X0	
	W0

El resultado de la función se devuelve por X0, y main() devuelve 0, así que así se prepara el valor de retorno. ¿Por qué usar la parte de 32 bits? Porque el tipo *int* en ARM64, igual que en x86-64, sigue siendo de 32 bits para una mejor compatibilidad. Por lo tanto, si una función devuelve un *int* de 32 bits, solo hay que llenar los 32 bits bajos del registro X0.

Para verificarlo, modifiquemos un poco este ejemplo y recompilémoslo. Ahora main() devuelve un valor de 64 bits:

Listing 3.16: main() que devuelve un valor de tipo uint64_t

```
#include <stdio.h>
#include <stdint.h>

uint64_t main()
{
    printf ("Hello!\n");
```

²²Frame Pointer

```

    return 0;
}

```

El resultado es el mismo, solo que así se ve ahora el MOV en esa línea:

Listing 3.17: Sin optimización GCC 4.8.1 + objdump

4005a4:	d2800000	mov	x0, #0x0	z
	// #0			

Luego, con la instrucción LDP (*Load Pair*) se restauran los registros X29 y X30. No hay signo de exclamación tras la instrucción: esto implica que primero se cargan los valores de la pila y solo después **SP** se incrementa en 16. Esto se llama *post-index*.

En ARM64 apareció una instrucción nueva: RET. Funciona igual que BX LR, , solo que se añade un bit de *pista* especial que informa a la **CPU** de que es un retorno de función, no un salto cualquiera, para poder ejecutarla de forma más óptima.

Debido a la simplicidad de la función, el GCC con optimización genera exactamente el mismo código.

3.5 MIPS

3.5.1

3.5.2 Con optimización GCC

Listing 3.18: Con optimización GCC 4.4.5 (salida de assembly)

```

1   $LC0:
2   ; \000 :
3   .ascii  "Hello, world!\012\000"
4
5   main:
6   ;
7   ;  GP:
8   lui      $28,%hi(__gnu_local_gp)
9   addiu   $sp,$sp,-32
10  addiu   $28,$28,%lo(__gnu_local_gp)
11  ;
12  sw      $31,28($sp)
13  ; $25:
14  lw      $25,%call16(puts)($28)
15  ; $4 ($a0):
16  lui      $4,%hi($LC0)
17  ;
18  jalr   $25

```

```

19      addiu   $4,$4,%lo($LC0) ; branch delay slot
20 ; RA:
21      lw       $31,28($sp)
22 ; $zero  $v0:
23      move    $2,$0
24 ; :
25      j       $31
26 ; :
27      addiu   $sp,$sp,32 ; branch delay slot

```

²³

, MOVE DST, SRC ADD DST, SRC, \$ZERO ($DST = SRC + 0$),.

Listing 3.19: Con optimización GCC 4.4.5 ([IDA](#))

```

1 .text:00000000 main:
2 .text:00000000
3 .text:00000000 var_10          = -0x10
4 .text:00000000 var_4           = -4
5 .text:00000000
6 ;
7 ;
8 ; GP:
9 .text:00000000              lui     $gp, (__gnu_local_gp >> 16)
10 .text:00000004             addiu   $sp, -0x20
11 .text:00000008             la      $gp, (__gnu_local_gp & 0xFFFF)
12 ;
13 .text:0000000C             sw      $ra, 0x20+var_4($sp)
14 ;
15 ;
16 .text:00000010             sw      $gp, 0x20+var_10($sp)
17 ; $t9:
18 .text:00000014             lw      $t9, (puts & 0xFFFF)($gp)
19 ; $a0:
20 .text:00000018             lui     $a0, ($LC0 >> 16) # "Hello, world!"
21 ;
22 .text:0000001C             jalr   $t9, $a0, ($LC0 & 0xFFFF) # "Hello, world!"
23 .text:00000020
24 ; RA:

```

²³[C.1 on page 1251](#)

```

25 .text:00000024          lw      $ra, 0x20+var_4($sp)
26 ; $zero $v0:
27 .text:00000028          move   $v0, $zero
28 ;
29 .text:0000002C          jr     $ra
30 ;
31 .text:00000030          addiu $sp, 0x20

```

3.5.3 Sin optimización GCC

Sin optimización GCC .

Listing 3.20: Sin optimización GCC 4.4.5 (salida de assembly)

```

1  $LC0:
2      .ascii  "Hello, world!\012\000"
3
4 main:
5 ;
6 ;
7      addiu $sp,$sp,-32
8      sw    $31,28($sp)
9      sw    $fp,24($sp)
10 ;
11     move  $fp,$sp
12 ; GP:
13     lui   $28,%hi(__gnu_local_gp)
14     addiu $28,$28,%lo(__gnu_local_gp)
15 ;
16     lui   $2,%hi($LC0)
17     addiu $4,$2,%lo($LC0)
18 ;
19     lw    $2,%call16(puts)($28)
20     nop
21 ; puts():
22     move  $25,$2
23     jalr  $25
24     nop  ; branch delay slot
25 ;
26     lw    $28,16($fp)
27 ; $2 ($V0) :
28     move  $2,$0
29 ;
30 ; SP:
31     move  $sp,$fp
32 ;
33 ; RA:

```

```

34      lw      $31,28($sp)
35 ;   FP:
36      lw      $fp,24($sp)
37      addiu $sp,$sp,32
38 ;   RA:
39      j      $31
40      nop   ; branch delay slot

```

Listing 3.21: Sin optimización GCC 4.4.5 ([IDA](#))

```

1 .text:00000000 main:
2 .text:00000000
3 .text:00000000 var_10          = -0x10
4 .text:00000000 var_8           = -8
5 .text:00000000 var_4           = -4
6 .text:00000000
7 ;
8 ;
9 ;
10 .text:00000000                addiu  $sp, -0x20
11 .text:00000004                sw     $ra, 0x20+var_4($sp)
12 .text:00000008                sw     $fp, 0x20+var_8($sp)
13 ;
14 .text:0000000C                move   $fp, $sp
15 ;   GP:
16 .text:00000010                la     $gp, __gnu_local_gp
17 .text:00000018                sw     $gp, 0x20+var_10($sp)
18 ;
19 .text:0000001C                lui    $v0, (aHelloWorld >> 16)|
20     ↴ # "Hello, world!"        addiu $a0, $v0, (aHelloWorld &|
21     ↴ 0xFFFF) # "Hello, world!"|
22 ;
23 .text:00000024                lw     $v0, (puts & 0xFFFF)($gp||
24     ↴ )|
25 .text:00000028                or     $at, $zero ; NOP
26 ;   puts():
27 .text:0000002C                move  $t9, $v0
28 .text:00000030                jalr $t9
29 .text:00000034                or     $at, $zero ; NOP
30 ;
31 .text:00000038                lw     $gp, 0x20+var_10($fp)
32 ;   $2 ($V0) :
33 .text:0000003C                move  $v0, $zero
34 ;
35 ;   SP:

```

```

34 .text:00000040          move   $sp, $fp
35 ; RA:
36 .text:00000044          lw      $ra, 0x20+var_4($sp)
37 ; FP:
38 .text:00000048          lw      $fp, 0x20+var_8($sp)
39 .text:0000004C          addiu $sp, 0x20
40 ; RA:
41 .text:00000050          jr      $ra
42 .text:00000054          or      $at, $zero ; NOP

```

OR \$AT, \$ZERO.

3.5.4

[2.3 on page 8.](#)

3.5.5 Con optimización GCC: GDB

Listing 3.22:

```

root@debian-mips:~# gcc hw.c -O3 -o hw
root@debian-mips:~# gdb hw
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
  ↳ licenses/gpl.html>
This is free software: you are free to change and redistribute ↳
  ↳ it.
There is NO WARRANTY, to the extent permitted by law. Type "↳
  ↳ show copying"
and "show warranty" for details.
This GDB was configured as "mips-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/hw...(no debugging symbols found)... ↳
  ↳ done.
(gdb) b main
Breakpoint 1 at 0x400654
(gdb) run
Starting program: /root/hw

Breakpoint 1, 0x00400654 in main ()
(gdb) set step-mode on
(gdb) disas
Dump of assembler code for function main:
0x00400640 <main+0>:    lui      gp,0x42

```

```

0x00400644 <main+4>:    addiu   sp,sp,-32
0x00400648 <main+8>:    addiu   gp,gp,-30624
0x0040064c <main+12>:   sw      ra,28(sp)
0x00400650 <main+16>:   sw      gp,16(sp)
0x00400654 <main+20>:   lw      t9,-32716(gp)
0x00400658 <main+24>:   lui     a0,0x40
0x0040065c <main+28>:   jalr   t9
0x00400660 <main+32>:   addiu  a0,a0,2080
0x00400664 <main+36>:   lw      ra,28(sp)
0x00400668 <main+40>:   move   v0,zero
0x0040066c <main+44>:   jr     ra
0x00400670 <main+48>:   addiu  sp,sp,32
End of assembler dump.
(gdb) s
0x00400658 in main ()
(gdb) s
0x0040065c in main ()
(gdb) s
0x2ab2de60 in printf () from /lib/libc.so.6
(gdb) x/s $a0
0x400820:          "hello, world"
(gdb)

```

3.6 Conclusión

La principal diferencia entre el código x86/ARM y x64/ARM64 es que el puntero a la cadena ahora es de 64 bits. En efecto, las CPU modernas son de 64 bits porque la memoria se abarató y ahora se puede instalar mucha más; para direccionarla, 32 bits ya no bastan. Por eso todos los punteros son ahora de 64 bits.

3.7 Ejercicios

3.7.1 Ejercicio #1

```

main:
    push 0xFFFFFFFF
    call MessageBeep
    xor  eax,eax
    retn

```

?¿Qué hace esta función win32?

Responda: [G.1.1 on page 1259](#).

3.7.2 Ejercicio #2

```
main:  
    pushq  %rbp  
    movq    %rsp, %rbp  
    movl    $2, %edi  
    call    sleep  
    popq    %rbp  
    ret
```

?¿Qué hace esta función Linux? Aquí se usa la sintaxis de ensamblador AT&T. Aquí se usa la sintaxis de ensamblador AT&T.

Responda: [G.1.1 on page 1259](#).

Capítulo 4

```
push    ebp  
mov     ebp, esp  
sub    esp, X
```

```
mov    esp, ebp  
pop    ebp  
ret    0
```

4.1 Recursión

: 36.3 on page 606.

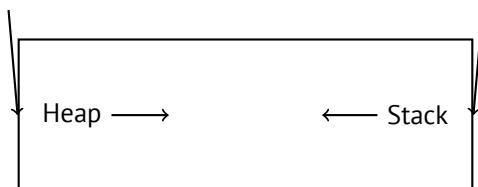
Capítulo 5

Pila

¹.

STMFD/LDMFD, STMED²/LDMED³ STMFA⁴/LDMFA⁵, STMEA⁶/LDMEA⁷

5.1



[RT74] :

¹[wikipedia.org/wiki/Call_stack](https://en.wikipedia.org/wiki/Call_stack)

²Store Multiple Empty Descending ()

³Load Multiple Empty Descending ()

⁴Store Multiple Full Ascending ()

⁵Load Multiple Full Ascending ()

⁶Store Multiple Empty Ascending ()

⁷Load Multiple Empty Ascending ()

The user-core part of an image is divided into three logical segments. The program text segment begins at location 0 in the virtual address space. During execution, this segment is write-protected and a single copy of it is shared among all processes executing the same program. At the first 8K byte boundary above the program text segment in the virtual address space begins a nonshared, writable data segment, the size of which may be extended by a system call. Starting at the highest address in the virtual address space is a stack segment, which automatically grows downward as the hardware's stack pointer fluctuates.

5.2

5.2.1

x86

```
void f()
{
    f();
}
```

```
c:\tmp6>cl ss.cpp /Fass.asm
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
ss.cpp
c:\tmp6\ss.cpp(4) : warning C4717: 'f' : recursive on all control paths, function will cause runtime stack overflow
```

...:

```
?f@@YAXXZ PROC ; f
; File c:\tmp6\ss.cpp
; Line 2
    push    ebp
    mov     ebp, esp
; Line 3
    call    ?f@@YAXXZ ; f
; Line 4
    pop    ebp
    ret    0
```

?f@@YAXXZ ENDP

; f

...

```
?f@@YAXXZ PROC ; f
; File c:\tmp6\ss.cpp
; Line 2
$LL3@f:
; Line 3
    jmp      SHORT $LL3@f
?f@@YAXXZ ENDP ; f
```

ARM

. «Hello, world!» (3.4 on page 17), LR (link register). . PUSH R4-R7,LR POP R4-R7,PC LR.

*leaf function*⁸. . ⁹.

8.3.2 on page 122, 8.3.3 on page 122, 19.17 on page 401, 19.33 on page 423, 19.5.4 on page 423, 15.4 on page 254, 15.2 on page 252, 17.3 on page 278.

5.2.2

«cdecl»:

```
push arg3
push arg2
push arg1
call f
add esp, 12 ; 4*3=12
```

ESP	
ESP+4	argumento#1, Marcado en IDA como arg_0
ESP+8	argumento#2, Marcado en IDA como arg_4
ESP+0xC	argumento#3, Marcado en IDA como arg_8
...	...

(64 on page 869).

10.

⁸infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka13785.html

⁹[Ray03, Chapter 4, Part II].

10

```
printf("%d %d %d", 1234);
```

```
printf()
```

```
: main(),main(int argc, char *argv[]) main(int argc, char *argv[],  
char *envp[]).
```

```
push envp  
push argv  
push argc  
call main  
...
```

```
main(int argc, char *argv[]), main(int argc),.
```

5.2.3

5.2.4 x86:

[11](#).

```
#ifdef __GNUC__  
#include <alloca.h> // GCC  
#else  
#include <malloc.h> // MSVC  
#endif  
#include <stdio.h>  
  
void f()  
{  
    char *buf=(char*)alloca (600);  
#ifdef __GNUC__  
    sprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // GCC  
#else  
    _snprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // MSVC  
#endif  
  
    puts (buf);  
};
```

MSVC

(MSVC 2010):

¹¹alloca16.asm y chkstk.asm en C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\crt\src\intel

Listing 5.1: MSVC 2010

```

...
    mov    eax, 600          ; 00000258H
    call   __alloca_probe_16
    mov    esi, esp

    push   3
    push   2
    push   1
    push   OFFSET $SG2672
    push   600              ; 00000258H
    push   esi
    call   __snprintf

    push   esi
    call   _puts
    add    esp, 28          ; 00000001cH

...

```

[12](#).

GCC + Sintaxis Intel

Listing 5.2: GCC 4.7.3

```

.LC0:
    .string "hi! %d, %d, %d\n"
f:
    push   ebp
    mov    ebp, esp
    push   ebx
    sub    esp, 660
    lea    ebx, [esp+39]
    and    ebx, -16
    mov    DWORD PTR [esp], ebx
    mov    DWORD PTR [esp+20], 3
    mov    DWORD PTR [esp+16], 2
    mov    DWORD PTR [esp+12], 1
    mov    DWORD PTR [esp+8], OFFSET FLAT:.LC0 ; "hi! %d, %
    ↵ %d, %d\n"
    mov    DWORD PTR [esp+4], 600           ; maxlen

```

[12](#).

```

call    _snprintf
mov     DWORD PTR [esp], ebx
call    puts
mov     ebx, DWORD PTR [ebp-4]
leave
ret

```

GCC + Sintaxis AT&T

:

Listing 5.3: GCC 4.7.3

```

.LC0:
.string "hi! %d, %d, %d\n"
f:
pushl  %ebp
movl   %esp, %ebp
pushl  %ebx
subl   $660, %esp
leal   39(%esp), %ebx
andl   $-16, %ebx
movl   %ebx, (%esp)
movl   $3, 20(%esp)
movl   $2, 16(%esp)
movl   $1, 12(%esp)
movl   $.LC0, 8(%esp)
movl   $600, 4(%esp)
call   _snprintf
movl   %ebx, (%esp)
call   puts
movl   -4(%ebp), %ebx
leave
ret

```

, movl \$3, 20(%esp) mov DWORD PTR [esp+20], 3

5.2.5 (Windows) SEH

: [\(68.3 on page 907\)](#).

5.2.6

[\(18.2 on page 334\)](#).

5.2.7**5.3**

...	...
ESP-0xC	#2, Marcado en IDA como var_8
ESP-8	#1, Marcado en IDA como var_4
ESP-4	EBP
ESP	
ESP+4	argumento#1, Marcado en IDA como arg_0
ESP+8	argumento#2, Marcado en IDA como arg_4
ESP+0xC	argumento#3, Marcado en IDA como arg_8
...	...

5.4

? :

```
#include <stdio.h>

void f1()
{
    int a=1, b=2, c=3;
};

void f2()
{
    int a, b, c;
    printf ("%d, %d, %d\n", a, b, c);
};

int main()
{
    f1();
    f2();
}
```

...

Listing 5.4: Sin optimización MSVC 2010

\$SG2752 DB	'%d, %d, %d', 0aH, 00H
_c\$ = -12	; size = 4
_b\$ = -8	; size = 4
_a\$ = -4	; size = 4

```

_f1    PROC
        push    ebp
        mov     ebp, esp
        sub     esp, 12
        mov     DWORD PTR _a$[ebp], 1
        mov     DWORD PTR _b$[ebp], 2
        mov     DWORD PTR _c$[ebp], 3
        mov     esp, ebp
        pop    ebp
        ret    0
_f1    ENDP

_c$ = -12      ; size = 4
_b$ = -8       ; size = 4
_a$ = -4       ; size = 4

_f2    PROC
        push    ebp
        mov     ebp, esp
        sub     esp, 12
        mov     eax, DWORD PTR _c$[ebp]
        push   eax
        mov     ecx, DWORD PTR _b$[ebp]
        push   ecx
        mov     edx, DWORD PTR _a$[ebp]
        push   edx
        push   OFFSET $SG2752 ; '%d, %d, %d'
        call   DWORD PTR __imp__printf
        add    esp, 16
        mov     esp, ebp
        pop    ebp
        ret    0
_f2    ENDP

_main  PROC
        push    ebp
        mov     ebp, esp
        call   _f1
        call   _f2
        xor    eax, eax
        pop    ebp
        ret    0
_main  ENDP

```

...

```

c:\Polygon\c>cl st.c /Fast.asm /MD
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00.40219.01 for 80x86

```

```
Copyright (C) Microsoft Corporation. All rights reserved.

st.c
c:\polygon\c\st.c(11) : warning C4700: uninitialized local ↴
    ↴ variable 'c' used
c:\polygon\c\st.c(11) : warning C4700: uninitialized local ↴
    ↴ variable 'b' used
c:\polygon\c\st.c(11) : warning C4700: uninitialized local ↴
    ↴ variable 'a' used
Microsoft (R) Incremental Linker Version 10.00.40219.01
Copyright (C) Microsoft Corporation. All rights reserved.

/out:st.exe
st.obj
```

...

```
c:\Polygon\c>st
1, 2, 3
```

f2().

OllyDbg:

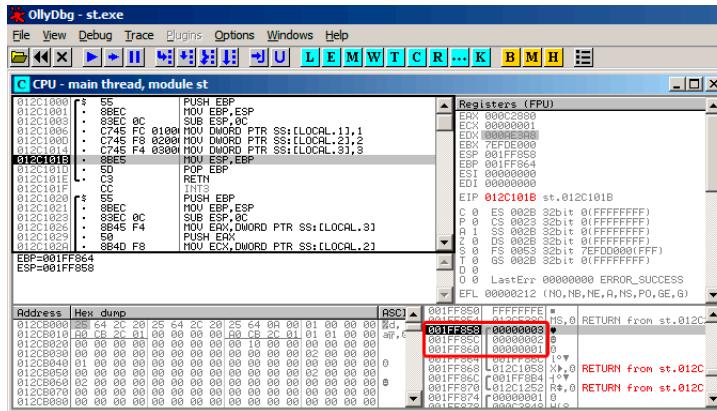


Figura 5.1: OllyDbg: f1()

`f1() a, b y c 0x14F860`

f2():

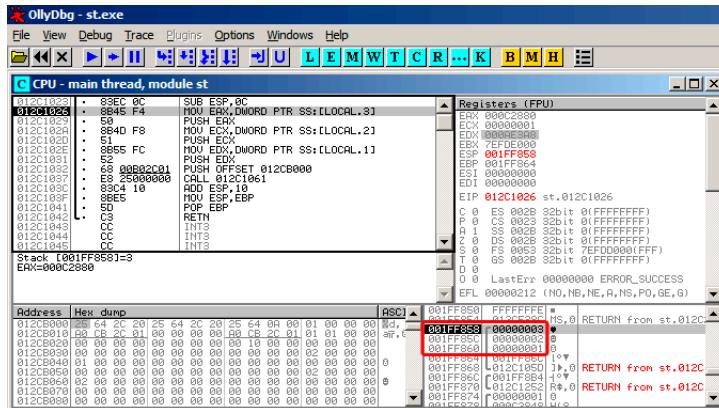


Figura 5.2: OllyDbg: f2()

... a, b y c f2()

?

5.5 Ejercicios

5.5.1 Ejercicio #1

?

```
#include <stdio.h>

int main()
{
    printf ("%d, %d, %d\n");

    return 0;
}
```

Responda: [G.1.2 on page 1259](#).

5.5.2 Ejercicio #2

Lo que hace el código?

Listing 5.5: Con optimización MSVC 2010

```
$SG3103 DB      '%d', 0aH, 00H

_main PROC
    push    0
    call    DWORD PTR __imp__time64
    push    edx
    push    eax
    push    OFFSET $SG3103 ; '%d'
    call    DWORD PTR __imp__printf
    add     esp, 16
    xor     eax, eax
    ret     0
_main ENDP
```

Listing 5.6: Con optimización Keil 6/2013 (Modo ARM)

```
main PROC
    PUSH    {r4,lr}
    MOV     r0,#0
    BL     time
    MOV     r1,r0
    ADR     r0,|L0.32|
    BL     __2printf
    MOV     r0,#0
    POP     {r4,pc}
    ENDP

|L0.32|
    DCB     "%d\n",0
```

Listing 5.7: Con optimización Keil 6/2013 (Modo Thumb)

```
main PROC
    PUSH    {r4,lr}
    MOVS   r0,#0
    BL     time
    MOVS   r1,r0
    ADR     r0,|L0.20|
    BL     __2printf
    MOVS   r0,#0
    POP     {r4,pc}
    ENDP

|L0.20|
    DCB     "%d\n",0
```

Listing 5.8: Con optimización GCC 4.9 (ARM64)

```

main:
    stp      x29, x30, [sp, -16]!
    mov      x0, 0
    add      x29, sp, 0
    bl       time
    mov      x1, x0
    ldp      x29, x30, [sp], 16
    adrp    x0, .LC0
    add      x0, x0, :lo12:.LC0
    b       printf
.LC0:
    .string "%d\n"

```

Listing 5.9: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

main:
var_10          = -0x10
var_4           = -4

        lui      $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x20
        la       $gp, (__gnu_local_gp & 0xFFFF)
        sw       $ra, 0x20+var_4($sp)
        sw       $gp, 0x20+var_10($sp)
        lw       $t9, (time & 0xFFFF)($gp)
        or       $at, $zero
        jalr    $t9
        move    $a0, $zero
        lw       $gp, 0x20+var_10($sp)
        lui      $a0, ($LC0 >> 16) # "%d\n"
        lw       $t9, (printf & 0xFFFF)($gp)
        lw       $ra, 0x20+var_4($sp)
        la       $a0, ($LC0 & 0xFFFF) # "%d\n"
        move    $a1, $v0
        jr       $t9
        addiu   $sp, 0x20

$LC0:         .ascii "%d\n" <0>           # DATA XREF: main+28

```

Responda: [G.1.2 on page 1260.](#)

Capítulo 6

printf() con varios argumentos

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d", 1, 2, 3);
    return 0;
}
```

6.1 x86

6.1.1 x86:

MSVC

```
$SG3830 DB      'a=%d; b=%d; c=%d', 00H
...
push    3
push    2
push    1
push    OFFSET $SG3830
call    _printf
add    esp, 16
        ; ↴ 00000010H
```

(64 on page 869).

```
push a1  
push a2  
call ...  
...  
push a1  
call ...  
...  
push a1  
push a2  
push a3  
call ...  
add esp, 24
```

Listing 6.1: x86

.text:100113E7	push	3
.text:100113E9	call	sub_100018B0 ; (3)
.text:100113EE	call	sub_100019D0 ;
.text:100113F3	call	sub_10006A90 ;
.text:100113F8	push	1
.text:100113FA	call	sub_100018B0 ; (1)
.text:100113FF	add	esp, 8 ;

MSVC y OllyDbg

OllyDbg.. MSVC 2012 /MD MSVCR*.DLL,

OllyDbg. ntdll.dll, F9(). CRT-. main().

•

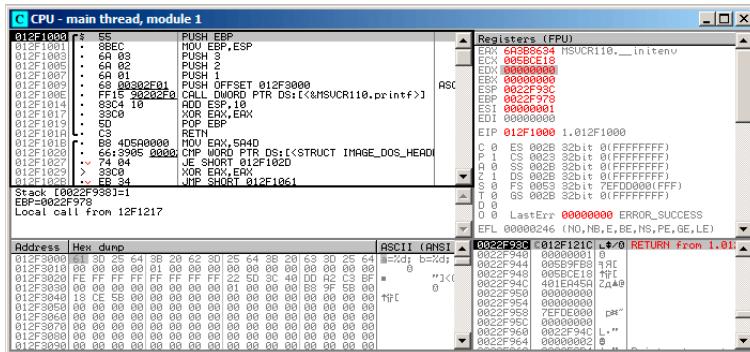


Figura 6.1: OllyDbg: main()

PUSH EBP F2 () F9 () . .

F8 (pasar por encima) 6 :

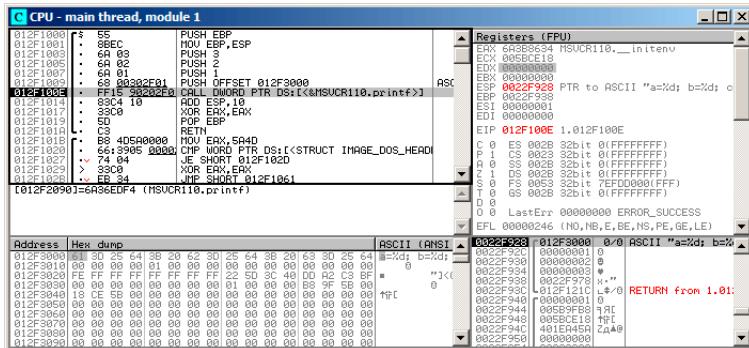


Figura 6.2: OllyDbg: printf()

PC CALL printf. OllyDbg, F8, EIP. ESP.

?

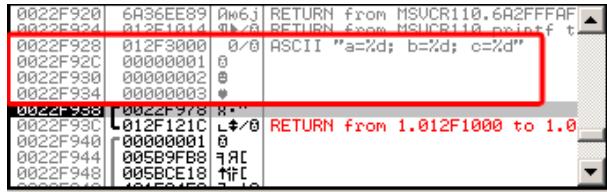


Figura 6.3: OllyDbg: ()

. OllyDbg printf(), .

• • • •

F8 (pasar por encima).

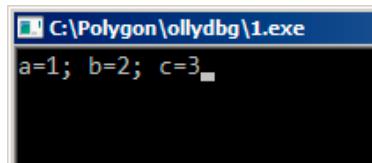


Figura 6.4: printf()

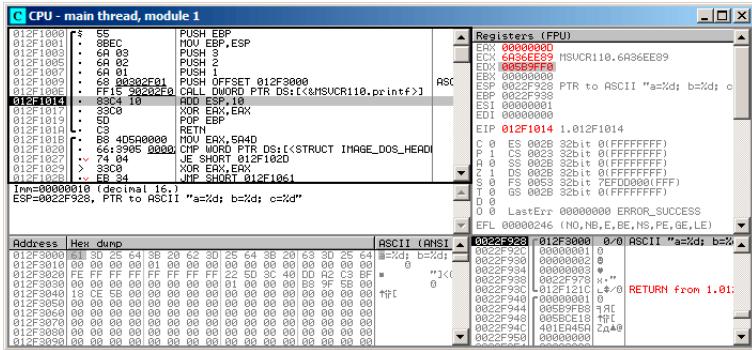


Figura 6.5: OllyDbg printf()

EAX 0xD (13). ECX y EDX..

F8 ADD ESP, 10:

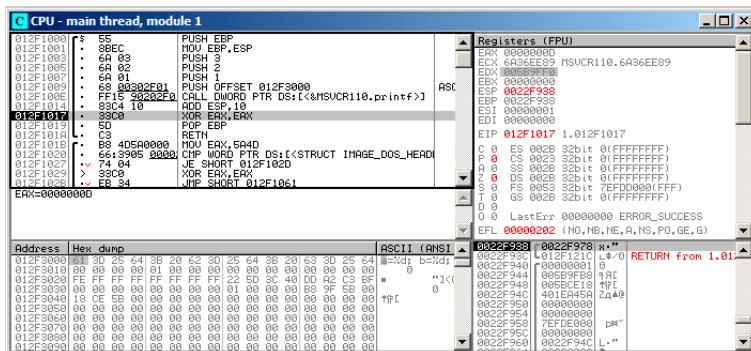


Figura 6.6: OllyDbg: ADD ESP, 10

ESP! (SP) o Basura..

GCC

```

main          proc near

var_10        = dword ptr -10h
var_C         = dword ptr -0Ch
var_8         = dword ptr -8
var_4         = dword ptr -4

        push    ebp
        mov     ebp, esp
        and     esp, 0FFFFFFF0h
        sub     esp, 10h
        mov     eax, offset aADBDCCD ; "a=%d; b=%d; c=%d\n"
        mov     [esp+10h+var_4], 3
        mov     [esp+10h+var_8], 2
        mov     [esp+10h+var_C], 1
        mov     [esp+10h+var_10], eax
        call    _printf
        mov     eax, 0
        leave
        retn
main          endp

```

GCC y GDB

-g .

```
$ gcc 1.c -g -o 1
```

```
$ gdb 1
GNU gdb (GDB) 7.6.1-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/遵守
    ↳ licenses/gpl.html>
This is free software: you are free to change and redistribute ↳
    ↳ it.
There is NO WARRANTY, to the extent permitted by law. Type "↙
    ↳ show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/dennis/polygon/1...done.
```

Listing 6.2: printf()

```
(gdb) b printf
Breakpoint 1 at 0x80482f0
```

. printf().

```
(gdb) run
Starting program: /home/dennis/polygon/1

Breakpoint 1, __printf (format=0x80484f0 "a=%d; b=%d; c=%d") at ↳
    ↳ printf.c:29
29      printf.c: No such file or directory.
```

(gdb) x/10w \$esp			
0xbffff11c:	0x0804844a	0x080484f0	0x00000001 ↳
↳	0x00000002		
0xbffff12c:	0x00000003	0x08048460	0x00000000 ↳
↳	0x00000000		
0xbffff13c:	0xb7e29905	0x00000001	

RA¹ (0x0804844a) :

¹Dirección de retorno

```
(gdb) x/5i 0x0804844a
0x804844a <main+45>: mov     $0x0,%eax
0x804844f <main+50>: leave
0x8048450 <main+51>: ret
0x8048451:   xchg    %ax,%ax
0x8048453:   xchg    %ax,%ax
```

```
(gdb) x/s 0x080484f0
0x80484f0: "a=%d; b=%d; c=%d"
```

(1, 2, 3) printf(). .

«finish».. printf().

```
(gdb) finish
Run till exit from #0  __printf (format=0x80484f0 "a=%d; b=%d; \n
    ↴ c=%d") at printf.c:29
main () at 1.c:6
6             return 0;
Value returned is $2 = 13
```

[GDB²](#) printf() EAX (13). .

«return 0;» . . ? . GDB .

. 13 en EAX:

```
(gdb) info registers
eax          0xd      13
ecx          0x0      0
edx          0x0      0
ebx 0xb7fc0000      -1208221696
esp 0xbfffff120      0xbfffff120
ebp 0xbfffff138      0xbfffff138
esi          0x0      0
edi          0x0      0
eip 0x804844a <main+45>
...
...
```

```
(gdb) disas
Dump of assembler code for function main:
0x0804841d <+0>: push    %ebp
0x0804841e <+1>:   mov     %esp,%ebp
```

²GNU debugger

```

0x08048420 <+3>:    and    $0xfffffffff0,%esp
0x08048423 <+6>:    sub    $0x10,%esp
0x08048426 <+9>:    movl   $0x3,0xc(%esp)
0x0804842e <+17>:   movl   $0x2,0x8(%esp)
0x08048436 <+25>:   movl   $0x1,0x4(%esp)
0x0804843e <+33>:   movl   $0x80484f0,(%esp)
0x08048445 <+40>:   call   0x80482f0 <printf@plt>
=> 0x0804844a <+45>:  mov    $0x0,%eax
0x0804844f <+50>:   leave 
0x08048450 <+51>:   ret    
End of assembler dump.

```

GDB AT&T.:

```

(gdb) set disassembly-flavor intel
(gdb) disas
Dump of assembler code for function main:
0x0804841d <+0>:    push   ebp
0x0804841e <+1>:    mov    ebp,esp
0x08048420 <+3>:    and    esp,0xfffffffff0
0x08048423 <+6>:    sub    esp,0x10
0x08048426 <+9>:    mov    DWORD PTR [esp+0xc],0x3
0x0804842e <+17>:   mov    DWORD PTR [esp+0x8],0x2
0x08048436 <+25>:   mov    DWORD PTR [esp+0x4],0x1
0x0804843e <+33>:   mov    DWORD PTR [esp],0x80484f0
0x08048445 <+40>:   call   0x80482f0 <printf@plt>
=> 0x0804844a <+45>:  mov    eax,0x0
0x0804844f <+50>:   leave 
0x08048450 <+51>:   ret    
End of assembler dump.

```

. GDB

```

(gdb) step
7      ;

```

MOV EAX, 0. EAX.

```

(gdb) info registers
eax          0x0      0
ecx          0x0      0
edx          0x0      0
ebx          0xb7fc0000      -1208221696
esp          0xbfffff120      0xbfffff120
ebp          0xbfffff138      0xbfffff138
esi          0x0      0
edi          0x0      0

```

eip	0x804844f	0x804844f <main+50>
...		

6.1.2 x64:

:

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n",
        1, 2, 3, 4, 5, 6, 7, 8);
    return 0;
}
```

MSVC

RCX, RDX, R8, R9 ...

Listing 6.3: MSVC 2012 x64

```
$SG2923 DB      'a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n'
        ', 0aH, 00H

main PROC
    sub    rsp, 88

    mov    DWORD PTR [rsp+64], 8
    mov    DWORD PTR [rsp+56], 7
    mov    DWORD PTR [rsp+48], 6
    mov    DWORD PTR [rsp+40], 5
    mov    DWORD PTR [rsp+32], 4
    mov    r9d, 3
    mov    r8d, 2
    mov    edx, 1
    lea    rcx, OFFSET FLAT:$SG2923
    call   printf

    ; 0
    xor    eax, eax

    add    rsp, 88
    ret    0
main ENDP
```

```
_TEXT    ENDS
END
```

GCC

RDI, RSI, RDX, RCX, R8, R9. . : [3.2.2 on page 15.](#)

: [3.2.2 on page 15.](#)

Listing 6.4: Con optimización GCC 4.4.6 x64

```
.LC0:
    .string "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    \n"

main:
    sub    rsp, 40

    mov    r9d, 5
    mov    r8d, 4
    mov    ecx, 3
    mov    edx, 2
    mov    esi, 1
    mov    edi, OFFSET FLAT:.LC0
    xor    eax, eax ;
    mov    DWORD PTR [rsp+16], 8
    mov    DWORD PTR [rsp+8], 7
    mov    DWORD PTR [rsp], 6
    call   printf

    ; 0

    xor    eax, eax
    add    rsp, 40
    ret
```

GCC + GDB

[GDB.](#)

```
$ gcc -g 2.c -o 2
```

```
$ gdb 2
GNU gdb (GDB) 7.6.1-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/dennis/polygon/2...done.
```

Listing 6.5:

```
(gdb) b printf
Breakpoint 1 at 0x400410
(gdb) run
Starting program: /home/dennis/polygon/2

Breakpoint 1, __printf (format=0x400628 "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n") at printf.c:29
29      printf.c: No such file or directory.
```

RSI/RDX/RCX/R8/R9 . RIP printf().

```
(gdb) info registers
rax          0x0      0
rbx          0x0      0
rcx          0x3      3
rdx          0x2      2
rsi          0x1      1
rdi          0x400628  4195880
rbp          0x7fffffffdf60 0x7fffffffdf60
rsp          0x7fffffffdf38 0x7fffffffdf38
r8           0x4      4
r9           0x5      5
r10          0x7fffffffdfce0 140737488346336
r11          0x7fffff7a65f60 140737348263776
r12          0x400440  4195392
r13          0x7ffffffffe040 140737488347200
r14          0x0      0
r15          0x0      0
rip          0x7fffff7a65f60 0x7fffff7a65f60 <__printf>
...
```

Listing 6.6:

```
(gdb) x/s $rdi
0x400628:    "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
               ↴ \n"
```

g giant words, .

(gdb) x/10g \$rsp	
0x7fffffffdf38: 0x0000000000400576	0x0000000000000006
0x7fffffffdf48: 0x0000000000000007	0x00007fff00000008
0x7fffffffdf58: 0x0000000000000000	0x0000000000000000
0x7fffffffdf68: 0x00007fffff7a33de5	0x0000000000000000
0x7fffffffdf78: 0x00007ffffffe048	0x0000000010000000

RA.: 6, 7, 8.: 0x00007fff00000008...

:

```
(gdb) set disassembly-flavor intel
(gdb) disas 0x0000000000400576
Dump of assembler code for function main:
0x000000000040052d <+0>:    push   rbp
0x000000000040052e <+1>:    mov    rbp,rs
0x0000000000400531 <+4>:    sub    rs,0x20
0x0000000000400535 <+8>:    mov    DWORD PTR [rs+0x10],0x8
0x000000000040053d <+16>:   mov    DWORD PTR [rs+0x8],0x7
0x0000000000400545 <+24>:   mov    DWORD PTR [rs],0x6
0x000000000040054c <+31>:   mov    r9d,0x5
0x0000000000400552 <+37>:   mov    r8d,0x4
0x0000000000400558 <+43>:   mov    ecx,0x3
0x000000000040055d <+48>:   mov    edx,0x2
0x0000000000400562 <+53>:   mov    esi,0x1
0x0000000000400567 <+58>:   mov    edi,0x400628
0x000000000040056c <+63>:   mov    eax,0x0
0x0000000000400571 <+68>:   call   0x400410 <printf@plt>
0x0000000000400576 <+73>:   mov    eax,0x0
0x000000000040057b <+78>:   leave 
0x000000000040057c <+79>:   ret
```

End of assembler dump.

.RIP LEAVE .

```
(gdb) finish
Run till exit from #0  __printf (format=0x400628 "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n") at printf.c:29
a=1; b=2; c=3; d=4; e=5; f=6; g=7; h=8
main () at 2.c:6
6                  return 0;
Value returned is $1 = 39
```

```
(gdb) next
7      };
(gdb) info registers
rax          0x0      0
rbx          0x0      0
rcx          0x26     38
rdx          0x7fffff7dd59f0  140737351866864
rsi          0x7fffffff9  2147483609
rdi          0x0      0
rbp          0x7fffffffdf60  0x7fffffffdf60
rsp          0x7fffffffdf40  0x7fffffffdf40
r8           0x7fffff7dd26a0  140737351853728
r9           0x7fffff7a60134  140737348239668
r10          0x7fffffff5b0  140737488344496
r11          0x7fffff7a95900  140737348458752
r12          0x400440 4195392
r13          0x7fffffff040  140737488347200
r14          0x0      0
r15          0x0      0
rip          0x40057b 0x40057b <main+78>
...

```

6.2 ARM

6.2.1 ARM:

- .fastcall ([64.3 on page 870](#)) o win64 ([64.5.1 on page 872](#)).

32- ARM

Sin optimización Keil 6/2013 (Modo ARM)

Listing 6.7: Sin optimización Keil 6/2013 (Modo ARM)

```
.text:00000000 main
.text:00000000 10 40 2D E9    STMFD   SP!, {R4,LR}
.text:00000004 03 30 A0 E3    MOV      R3, #3
.text:00000008 02 20 A0 E3    MOV      R2, #2
.text:0000000C 01 10 A0 E3    MOV      R1, #1
.text:00000010 08 00 8F E2    ADR      R0, aADBD_CD ; "a=%d; b=%d; c=%d"
.text:00000014 06 00 00 EB    BL       _2printf
.text:00000018 00 00 A0 E3    MOV      R0, #0 ; return 0
.text:0000001C 10 80 BD E8    LDMFD   SP!, {R4,PC}
```

0x18 0 R0 *return 0.*

Con optimización Keil 6/2013.

Con optimización Keil 6/2013 (Modo Thumb)

Listing 6.8: Con optimización Keil 6/2013 (Modo Thumb)

```
.text:00000000 main
.text:00000000 10 B5      PUSH    {R4,LR}
.text:00000002 03 23      MOVS    R3, #3
.text:00000004 02 22      MOVS    R2, #2
.text:00000006 01 21      MOVS    R1, #1
.text:00000008 02 A0      ADR     R0, aADBDLCD ; "a=%d; b=%d; c=%d"
.text:0000000A 00 F0 0D F8  BL      __2printf
.text:0000000E 00 20      MOVS    R0, #0
.text:00000010 10 BD      POP    {R4,PC}
```

Con optimización Keil 6/2013 (Modo ARM) + return

return 0:

```
#include <stdio.h>

void main()
{
    printf("a=%d; b=%d; c=%d", 1, 2, 3);
}
```

Listing 6.9: Con optimización Keil 6/2013 (Modo ARM)

```
.text:00000014 main
.text:00000014 03 30 A0 E3  MOV     R3, #3
.text:00000018 02 20 A0 E3  MOV     R2, #2
.text:0000001C 01 10 A0 E3  MOV     R1, #1
.text:00000020 1E 0E 8F E2  ADR     R0, aADBDLCD ; "a=%d; b=%d; c=%d\n"
.text:00000024 CB 18 00 EA  B      __2printf
```

.... .!.

: 13.1.1 on page 184.

ARM64**Sin optimización GCC (Linaro) 4.9**

Listing 6.10: Sin optimización GCC (Linaro) 4.9

```
.LC1:
    .string "a=%d; b=%d; c=%d"
f2:
; :
    stp    x29, x30, [sp, -16]!
; (FP=SP):
    add    x29, sp, 0
    adrp   x0, .LC1
    add    x0, x0, :lo12:.LC1
    mov    w1, 1
    mov    w2, 2
    mov    w3, 3
    bl     printf
    mov    w0, 0
;
    ldp    x29, x30, [sp], 16
    ret
```

Con optimización GCC (Linaro) 4.9 .

6.2.2 ARM:

[: 6.1.2 on page 55.](#)

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n",
    ↴ n", 1, 2, 3, 4, 5, 6, 7, 8);
    return 0;
}
```

Con optimización Keil 6/2013: Modo ARM

.text:00000028	main
.text:00000028	
.text:00000028	var_18 = -0x18
.text:00000028	var_14 = -0x14

```
.text:00000028          var_4    = -4
.text:00000028
.text:00000028 04 E0 2D E5  STR    LR, [SP,#var_4]!
.text:0000002C 14 D0 4D E2  SUB    SP, SP, #0x14
.text:00000030 08 30 A0 E3  MOV    R3, #8
.text:00000034 07 20 A0 E3  MOV    R2, #7
.text:00000038 06 10 A0 E3  MOV    R1, #6
.text:0000003C 05 00 A0 E3  MOV    R0, #5
.text:00000040 04 C0 8D E2  ADD    R12, SP, #0x18+var_14
.text:00000044 0F 00 8C E8  STMIA  R12, {R0-R3}
.text:00000048 04 00 A0 E3  MOV    R0, #4
.text:0000004C 00 00 8D E5  STR    R0, [SP,#0x18+var_18]
.text:00000050 03 30 A0 E3  MOV    R3, #3
.text:00000054 02 20 A0 E3  MOV    R2, #2
.text:00000058 01 10 A0 E3  MOV    R1, #1
.text:0000005C 6E 0F 8F E2  ADR    R0, aADBDCDDDEDFDGD ; "a=%d; \n
    ↴ b=%d; c=%d; d=%d; e=%d; f=%d; g=%"...
.text:00000060 BC 18 00 EB  BL     __2printf
.text:00000064 14 D0 8D E2  ADD    SP, SP, #0x14
.text:00000068 04 F0 9D E4  LDR    PC, [SP+4+var_4],#4
```

:

• :

STR LR, [SP,#var_4]!. *pre-index*. .: [28.2 on page 574](#).

SUB SP, SP, #0x14

• : . ADD R12, SP, #0x18+var_14 . *var_14*, -0x14 STMIA R12,
R0-R3 STMIA *Store Multiple Increment After*. «*Increment After*»

• : STR R0, [SP,#0x18+var_18] . *var_18* -0x18, ,.

• :

,:

• .

• :

ADD SP, SP, #0x14.

LDR PC, [SP+4+var_4],#4 (4 + *var_4* = 4 + (-4) = 0, LDR PC,
[SP],#4), SP 4. *post-index*³. IDA? *var_4* LR. POP PC en x86⁴.

³: [28.2 on page 574](#).

⁴:

Con optimización Keil 6/2013: Modo Thumb

```

.text:0000001C          printf_main2
.text:0000001C
.text:0000001C          var_18 = -0x18
.text:0000001C          var_14 = -0x14
.text:0000001C          var_8  = -8
.text:0000001C
.text:0000001C 00 B5    PUSH   {LR}
.text:0000001E 08 23    MOVS   R3, #8
.text:00000020 85 B0    SUB    SP, SP, #0x14
.text:00000022 04 93    STR    R3, [SP,#0x18+var_8]
.text:00000024 07 22    MOVS   R2, #7
.text:00000026 06 21    MOVS   R1, #6
.text:00000028 05 20    MOVS   R0, #5
.text:0000002A 01 AB    ADD    R3, SP, #0x18+var_14
.text:0000002C 07 C3    STMIA  R3!, {R0-R2}
.text:0000002E 04 20    MOVS   R0, #4
.text:00000030 00 90    STR    R0, [SP,#0x18+var_18]
.text:00000032 03 23    MOVS   R3, #3
.text:00000034 02 22    MOVS   R2, #2
.text:00000036 01 21    MOVS   R1, #1
.text:00000038 A0 A0    ADR    R0, aADBDCDDDEFDGD ; "a=%d\n
               ; b=%d; c=%d; d=%d; e=%d; f=%d; g=%"...
.text:0000003A 06 F0 D9 F8    BL     __printf
.text:0000003E
.text:0000003E          loc_3E ; CODE XREF: example13_f+16
.text:0000003E 05 B0    ADD    SP, SP, #0x14
.text:00000040 00 BD    POP    {PC}

```

Con optimización Xcode 4.6.3 (LLVM): Modo ARM

```

__text:0000290C          _printf_main2
__text:0000290C
__text:0000290C          var_1C = -0x1C
__text:0000290C          var_C  = -0xC
__text:0000290C
__text:0000290C 80 40 2D E9    STMFD  SP!, {R7,LR}
__text:00002910 0D 70 A0 E1    MOV    R7, SP
__text:00002914 14 D0 4D E2    SUB    SP, SP, #0x14
__text:00002918 70 05 01 E3    MOV    R0, #0x1570
__text:0000291C 07 C0 A0 E3    MOV    R12, #7
__text:00002920 00 00 40 E3    MOVT   R0, #0
__text:00002924 04 20 A0 E3    MOV    R2, #4
__text:00002928 00 00 8F E0    ADD    R0, PC, R0

```

```

_text:0000292C 06 30 A0 E3    MOV    R3, #6
_text:00002930 05 10 A0 E3    MOV    R1, #5
_text:00002934 00 20 8D E5    STR    R2, [SP,#0x1C+var_1C]
_text:00002938 0A 10 8D E9    STMFA  SP, {R1,R3,R12}
_text:0000293C 08 90 A0 E3    MOV    R9, #8
_text:00002940 01 10 A0 E3    MOV    R1, #1
_text:00002944 02 20 A0 E3    MOV    R2, #2
_text:00002948 03 30 A0 E3    MOV    R3, #3
_text:0000294C 10 90 8D E5    STR    R9, [SP,#0x1C+var_C]
_text:00002950 A4 05 00 EB    BL     _printf
_text:00002954 07 D0 A0 E1    MOV    SP, R7
_text:00002958 80 80 BD E8    LDMFD SP!, {R7,PC}

```

STMFA (Store Multiple Full Ascending) STMIB (Store Multiple Increment Before). .
.... MOVT R0, #0 y ADD R0, PC, R0. MOVT R0, #0 y MOV R2, #4 ..

Con optimización Xcode 4.6.3 (LLVM): Modo Thumb-2

```

_text:00002BA0           _printf_main2
_text:00002BA0
_text:00002BA0           var_1C = -0x1C
_text:00002BA0           var_18 = -0x18
_text:00002BA0           var_C = -0xC
_text:00002BA0
_text:00002BA0 80 B5      PUSH   {R7,LR}
_text:00002BA2 6F 46      MOV    R7, SP
_text:00002BA4 85 B0      SUB    SP, SP, #0x14
_text:00002BA6 41 F2 D8 20 MOVW   R0, #0x12D8
_text:00002BAA 4F F0 07 0C MOV.W  R12, #7
_text:00002BAE C0 F2 00 00 MOVT.W R0, #0
_text:00002BB2 04 22      MOVS   R2, #4
_text:00002BB4 78 44      ADD    R0, PC ; char *
_text:00002BB6 06 23      MOVS   R3, #6
_text:00002BB8 05 21      MOVS   R1, #5
_text:00002BBA 0D F1 04 0E ADD.W LR, SP, #0x1C+var_18
_text:00002BBE 00 92      STR    R2, [SP,#0x1C+var_1C]
_text:00002BC0 4F F0 08 09 MOV.W R9, #8
_text:00002BC4 8E E8 0A 10 STMIA.W LR, {R1,R3,R12}
_text:00002BC8 01 21      MOVS   R1, #1
_text:00002BCA 02 22      MOVS   R2, #2
_text:00002BCC 03 23      MOVS   R3, #3
_text:00002BCE CD F8 10 90 STR.W R9, [SP,#0x1C+var_C]
_text:00002BD2 01 F0 0A EA BLX   _printf
_text:00002BD6 05 B0      ADD    SP, SP, #0x14
_text:00002BD8 80 BD      POP    {R7,PC}

```

ARM64**Sin optimización GCC (Linaro) 4.9**

Listing 6.11: Sin optimización GCC (Linaro) 4.9

```

.LC2:
    .string "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    \n"
f3:
; :
    sub      sp, sp, #32
; :
    stp      x29, x30, [sp,16]
; (FP=SP):
    add      x29, sp, 16
    adrp    x0, .LC2 ; "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    \n"
    add      x0, x0, :lo12:.LC2
    mov      w1, 8           ; 9th argument
    str      w1, [sp]        ; store 9th argument in the stack
    \n"
    mov      w1, 1
    mov      w2, 2
    mov      w3, 3
    mov      w4, 4
    mov      w5, 5
    mov      w6, 6
    mov      w7, 7
    bl       printf
    sub      sp, x29, #16
;
    ldp      x29, x30, [sp,16]
    add      sp, sp, 32
    ret

```

:[[ARM13c](#)]. X0.

Con optimización GCC (Linaro) 4.9 .

6.3 MIPS

6.3.1 3

Con optimización GCC 4.4.5

\$5...\$7 (o \$A0...\$A2).

Listing 6.12: Con optimización GCC 4.4.5 (salida de assembly)

```

$LC0:
    .ascii  "a=%d; b=%d; c=%d\000"
main:
; :
    lui      $28,%hi(__gnu_local_gp)
    addiu   $sp,$sp,-32
    addiu   $28,$28,%lo(__gnu_local_gp)
    sw      $31,28($sp)
; printf():
    lw       $25,%call16(sprintf)($28)
; printf():
    lui      $4,%hi($LC0)
    addiu   $4,$4,%lo($LC0)
; printf():
    li       $5,1                      # 0x1
; printf():
    li       $6,2                      # 0x2
; printf():
    jalr    $25
; printf() (branch delay slot):
    li       $7,3                      # 0x3

; :
    lw       $31,28($sp)
; 0:
    move   $2,$0
;
    j      $31
    addiu $sp,$sp,32 ; branch delay slot

```

Listing 6.13: Con optimización GCC 4.4.5 (IDA)

```

.text:00000000 main:
.text:00000000
.text:00000000 var_10          = -0x10
.text:00000000 var_4           = -4

```

```

.text:00000000
; :
.text:00000000
    lui      $gp, (__gnu_local_gp >> 16)
.text:00000004
    addiu   $sp, -0x20
.text:00000008
    la      $gp, (__gnu_local_gp & 0xFFFF)
.text:0000000C
    sw      $ra, 0x20+var_4($sp)
.text:00000010
    sw      $gp, 0x20+var_10($sp)
; printf():
.text:00000014
    lw      $t9, (printf & 0xFFFF)(16)
; printf():
; printf():
.text:00000018
    la      $a0, $LC0          # "a=%d\n"
; printf():
; printf():
.text:00000020
    li      $a1, 1
; printf():
; printf():
.text:00000024
    li      $a2, 2
; printf():
; printf():
.text:00000028
    jalr   $t9
; printf() (branch delay slot):
.text:0000002C
    li      $a3, 3
; :
.text:00000030
    lw      $ra, 0x20+var_4($sp)
; 0:
.text:00000034
    move   $v0, $zero
;
.text:00000038
    jr      $ra
.text:0000003C
    addiu   $sp, 0x20 ; branch delay slot

```

Sin optimización GCC 4.4.5

Sin optimización GCC :

Listing 6.14: Sin optimización GCC 4.4.5 (salida de assembly)

```

$LC0:
    .ascii  "a=%d; b=%d; c=%d\000"
main:
; :
    addiu   $sp,$sp,-32
    sw      $31,28($sp)
    sw      $fp,24($sp)
    move   $fp,$sp

```

```

        lui      $28,%hi(__gnu_local_gp)
        addiu   $28,$28,%lo(__gnu_local_gp)
;
        lui      $2,%hi($LC0)
        addiu   $2,$2,%lo($LC0)
; printf():
        move    $4,$2
; printf():
        li      $5,1                      # 0x1
; printf():
        li      $6,2                      # 0x2
; printf():
        li      $7,3                      # 0x3
; printf():
        lw      $2,%call16(sprintf)($28)
        nop
; printf():
        move    $25,$2
        jalr   $25
        nop
;
        lw      $28,16($fp)
; 0:
        move    $2,$0
        move    $sp,$fp
        lw      $31,28($sp)
        lw      $fp,24($sp)
        addiu  $sp,$sp,32
;
        j      $31
        nop

```

Listing 6.15: Sin optimización GCC 4.4.5 (IDA)

```

.text:00000000 main:
.text:00000000
.text:00000000 var_10          = -0x10
.text:00000000 var_8           = -8
.text:00000000 var_4           = -4
.text:00000000
;
.text:00000000                 addiu   $sp, -0x20
.text:00000004                 sw      $ra, 0x20+var_4($sp)
.text:00000008                 sw      $fp, 0x20+var_8($sp)
.text:0000000C                 move    $fp, $sp
.text:00000010                 la      $gp, __gnu_local_gp

```

.text:00000018	sw	\$gp, 0x20+var_10(\$sp)
; :		
.text:0000001C	la	\$v0, aADBDLCD # "a=%d\n"
↳ ; b=%d; c=%d"		
; printf():		
.text:00000024	move	\$a0, \$v0
; printf():		
.text:00000028	li	\$a1, 1
; printf():		
.text:0000002C	li	\$a2, 2
; printf():		
.text:00000030	li	\$a3, 3
; printf():		
.text:00000034	lw	\$v0, (printf & 0xFFFF)(↳
↳ \$gp)		
.text:00000038	or	\$at, \$zero
; printf():		
.text:0000003C	move	\$t9, \$v0
.text:00000040	jalr	\$t9
.text:00000044	or	\$at, \$zero ; NOP
; :		
.text:00000048	lw	\$gp, 0x20+var_10(\$fp)
; 0:		
.text:0000004C	move	\$v0, \$zero
.text:00000050	move	\$sp, \$fp
.text:00000054	lw	\$ra, 0x20+var_4(\$sp)
.text:00000058	lw	\$fp, 0x20+var_8(\$sp)
.text:0000005C	addiu	\$sp, 0x20
;		
.text:00000060	jr	\$ra
.text:00000064	or	\$at, \$zero ; NOP

6.3.2 8

:6.1.2 on page 55.

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    ↳ n", 1, 2, 3, 4, 5, 6, 7, 8);
    return 0;
}
```

Con optimización GCC 4.4.5

Listing 6.16: Con optimización GCC 4.4.5 (salida de assembly)

```

$LC0:
    .ascii  "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    \012\000"
main:
; :
    lui      $28,%hi(__gnu_local_gp)
    addiu   $sp,$sp,-56
    addiu   $28,$28,%lo(__gnu_local_gp)
    sw      $31,52($sp)
; :
    li       $2,4                      # 0x4
    sw      $2,16($sp)
; :
    li       $2,5                      # 0x5
    sw      $2,20($sp)
; :
    li       $2,6                      # 0x6
    sw      $2,24($sp)
; :
    li       $2,7                      # 0x7
    lw      $25,%call16(sprintf)($28)
    sw      $2,28($sp)
; $a0:
    lui      $4,%hi($LC0)
; :
    li       $2,8                      # 0x8
    sw      $2,32($sp)
    addiu   $4,$4,%lo($LC0)
; $a1:
    li       $5,1                      # 0x1
; $a2:
    li       $6,2                      # 0x2
; printf():
    jalr   $25
; $a3 (branch delay slot):
    li       $7,3                      # 0x3

; :
    lw      $31,52($sp)
; 0:
    move   $2,$0
;
    j      $31

```

addiu \$sp,\$sp,56 ; branch delay slot
--

Listing 6.17: Con optimización GCC 4.4.5 (IDA)

```

.text:00000000 main:
.text:00000000
.text:00000000 var_28          = -0x28
.text:00000000 var_24          = -0x24
.text:00000000 var_20          = -0x20
.text:00000000 var_1C          = -0x1C
.text:00000000 var_18          = -0x18
.text:00000000 var_10          = -0x10
.text:00000000 var_4           = -4
.text:00000000
; :
.text:00000000                 lui    $gp, (__gnu_local_gp >> 16)
.text:00000004 addiu $sp, -0x38
.text:00000008 la    $gp, (__gnu_local_gp & 0xFFFF)
.text:0000000C sw    $ra, 0x38+var_4($sp)
.text:00000010 sw    $gp, 0x38+var_10($sp)
; :
.text:00000014 li    $v0, 4
.text:00000018 sw    $v0, 0x38+var_28($sp)
; :
.text:0000001C li    $v0, 5
.text:00000020 sw    $v0, 0x38+var_24($sp)
; :
.text:00000024 li    $v0, 6
.text:00000028 sw    $v0, 0x38+var_20($sp)
; :
.text:0000002C li    $v0, 7
.text:00000030 lw    $t9, (printf & 0xFFFF)(-$gp)
.text:00000034 sw    $v0, 0x38+var_1C($sp)
; $a0:
.text:00000038 lui   $a0, ($LC0 >> 16) # "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%"...
; :
.text:0000003C li    $v0, 8
.text:00000040 sw    $v0, 0x38+var_18($sp)
; $a1:
.text:00000044 la    $a0, ($LC0 & 0xFFFF) # "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%"...
; $a1:
.text:00000048 li    $a1, 1

```

```
; $a2:
.text:0000004C          li      $a2, 2
; printf():
.text:00000050          jalr    $t9
; $a3 (branch delay slot):
.text:00000054          li      $a3, 3
; :
.text:00000058          lw      $ra, 0x38+var_4($sp)
; 0:
.text:0000005C          move   $v0, $zero
;
.text:00000060          jr      $ra
.text:00000064          addiu  $sp, 0x38 ; branch delay
    ↳ slot
```

Sin optimización GCC 4.4.5

Sin optimización GCC :

Listing 6.18: Sin optimización GCC 4.4.5 (salida de assembly)

```
$LC0:
    .ascii  "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n"
    ↳ \012\000"
main:
; :
    addiu  $sp,$sp,-56
    sw     $31,52($sp)
    sw     $fp,48($sp)
    move   $fp,$sp
    lui    $28,%hi(__gnu_local_gp)
    addiu  $28,$28,%lo(__gnu_local_gp)
    lui    $2,%hi($LC0)
    addiu  $2,$2,%lo($LC0)
; :
    li     $3,4                  # 0x4
    sw     $3,16($sp)
; :
    li     $3,5                  # 0x5
    sw     $3,20($sp)
; :
    li     $3,6                  # 0x6
    sw     $3,24($sp)
; :
    li     $3,7                  # 0x7
    sw     $3,28($sp)
```

```

; :
    li      $3,8          # 0x8
    sw      $3,32($sp)
; $a0:
    move   $4,$2
; $a1:
    li      $5,1          # 0x1
; $a2:
    li      $6,2          # 0x2
; $a3:
    li      $7,3          # 0x3
; printf():
    lw      $2,%call16(sprintf)($28)
    nop
    move   $25,$2
    jalr   $25
    nop
; :
    lw      $28,40($fp)
; 0:
    move   $2,$0
    move   $sp,$fp
    lw      $31,52($sp)
    lw      $fp,48($sp)
    addiu $sp,$sp,56
;
    j      $31
    nop

```

Listing 6.19: Sin optimización GCC 4.4.5 (IDA)

```

.text:00000000 main:
.text:00000000
.text:00000000 var_28        = -0x28
.text:00000000 var_24        = -0x24
.text:00000000 var_20        = -0x20
.text:00000000 var_1C        = -0x1C
.text:00000000 var_18        = -0x18
.text:00000000 var_10        = -0x10
.text:00000000 var_8         = -8
.text:00000000 var_4         = -4
.text:00000000
; :
.text:00000000 addiu   $sp, -0x38
.text:00000004 sw      $ra, 0x38+var_4($sp)
.text:00000008 sw      $fp, 0x38+var_8($sp)
.text:0000000C move   $fp, $sp

```

CAPÍTULO 6. PRINTF() CON VARIOS ARGUMENTOS

```

.text:00000010      la    $gp, __gnu_local_gp
.text:00000018      sw    $gp, 0x38+var_10($sp)
.text:0000001C      la    $v0, aABDCDDDEDFDGD # 
    ↴ "a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%"...
; :
.text:00000024      li    $v1, 4
.text:00000028      sw    $v1, 0x38+var_28($sp)
; :
.text:0000002C      li    $v1, 5
.text:00000030      sw    $v1, 0x38+var_24($sp)
; :
.text:00000034      li    $v1, 6
.text:00000038      sw    $v1, 0x38+var_20($sp)
; :
.text:0000003C      li    $v1, 7
.text:00000040      sw    $v1, 0x38+var_1C($sp)
; :
.text:00000044      li    $v1, 8
.text:00000048      sw    $v1, 0x38+var_18($sp)
; $a0:
.text:0000004C      move $a0, $v0
; $a1:
.text:00000050      li    $a1, 1
; $a2:
.text:00000054      li    $a2, 2
; $a3:
.text:00000058      li    $a3, 3
; printf():
.text:0000005C      lw    $v0, (printf & 0xFFFF)(
    ↴ $gp)
.text:00000060      or    $at, $zero
.text:00000064      move $t9, $v0
.text:00000068      jalr $t9
.text:0000006C      or    $at, $zero ; NOP
; :
.text:00000070      lw    $gp, 0x38+var_10($fp)
; 0:
.text:00000074      move $v0, $zero
.text:00000078      move $sp, $fp
.text:0000007C      lw    $ra, 0x38+var_4($sp)
.text:00000080      lw    $fp, 0x38+var_8($sp)
.text:00000084      addiu $sp, 0x38
;
.text:00000088      jr    $ra
.text:0000008C      or    $at, $zero ; NOP

```

CAPÍTULO 6. PRINTF() CON VARIOS ARGUMENTOS

6.4 Conclusión

:

Listing 6.20: x86

```
...  
PUSH  
PUSH  
PUSH  
CALL  
;
```

Listing 6.21: x64 (MSVC)

```
MOV RCX,  
MOV RDX,  
MOV R8,  
MOV R9,  
...  
PUSH  
CALL  
;
```

Listing 6.22: x64 (GCC)

```
MOV RDI,  
MOV RSI,  
MOV RDX,  
MOV RCX,  
MOV R8,  
MOV R9,  
...  
PUSH  
CALL  
;
```

Listing 6.23: ARM

```
MOV R0,  
MOV R1,  
MOV R2,  
MOV R3,  
;
```

```
BL
```

```
;
```

Listing 6.24: ARM64

```
MOV X0,  
MOV X1,  
MOV X2,  
MOV X3,  
MOV X4,  
MOV X5,  
MOV X6,  
MOV X7,  
;  
BL CALL  
;
```

Listing 6.25: MIPS ()

```
LI $4, ; AKA $A0  
LI $5, ; AKA $A1  
LI $6, ; AKA $A2  
LI $7, ; AKA $A3  
;  
LW temp_reg,  
JALR temp_reg
```

6.5

CPU .

Capítulo 7

scanf()

7.1

```
#include <stdio.h>

int main()
{
    int x;
    printf ("Enter X:\n");
    scanf ("%d", &x);
    printf ("You entered %d...\n", x);
    return 0;
}
```

7.1.1

C/C++

; `memcpy()`, `void*`,
([\(10 on page 130\)](#)).`scanf()`.

C/C++

7.1.2 x86

MSVC

```
CONST    SEGMENT
$SG3831   DB      'Enter X:', 0aH, 00H
$SG3832   DB      '%d', 00H
$SG3833   DB      'You entered %d...', 0aH, 00H
CONST    ENDS
PUBLIC   _main
EXTRN   _scanf:PROC
EXTRN   _printf:PROC
; Function compile flags: /Odt
_TEXT    SEGMENT
_x$ = -4           ; size = 4
_main   PROC
    push   ebp
    mov    ebp, esp
    push   ecx
    push   OFFSET $SG3831 ; 'Enter X:'
    call   _printf
    add    esp, 4
    lea    eax, DWORD PTR _x$[ebp]
    push   eax
    push   OFFSET $SG3832 ; '%d'
    call   _scanf
    add    esp, 8
    mov    ecx, DWORD PTR _x$[ebp]
    push   ecx
    push   OFFSET $SG3833 ; 'You entered %d...'
    call   _printf
    add    esp, 8

    ; 0
    xor    eax, eax
    mov    esp, ebp
    pop    ebp
    ret    0
_main   ENDP
_TEXT    ENDS
```

:

...	...
EBP-8	#2, Marcado en IDA como var_8
EBP-4	#1, Marcado en IDA como var_4
EBP	EBP
EBP+4	
EBP+8	argumento#1, Marcado en IDA como arg_0
EBP+0xC	argumento#2, Marcado en IDA como arg_4
EBP+0x10	argumento#3, Marcado en IDA como arg_8
...	...

(A.6.2 on page 1233).

lea eax, [ebp-4].

You entered %d...\\n.

7.1.3 MSVC + OllyDbg

OllyDbg. ntdll.dll..

:

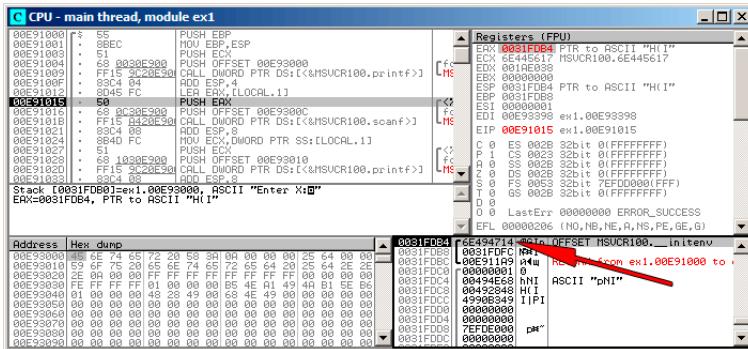


Figura 7.1: OllyDbg:

«Follow in stack». . . (0x6E494714) . . . :

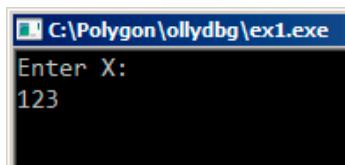


Figura 7.2:

scanf():

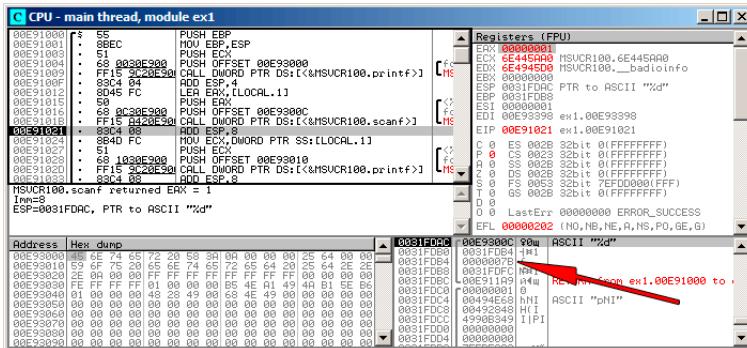


Figura 7.3: OllyDbg: scanf()

scanf() 1 en EAX, . 0x7B (123).

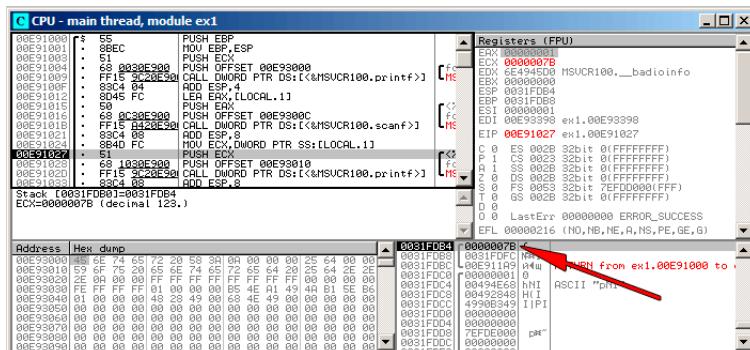


Figura 7.4: OllyDbg: printf()

GCC

```

main          proc near
var_20        = dword ptr -20h
var_1C        = dword ptr -1Ch
var_4         = dword ptr -4

push    ebp
mov     ebp, esp
and    esp, 0FFFFFFF0h
sub    esp, 20h
mov     [esp+20h+var_20], offset aEnterX ; "↙
↳ Enter X:"      _puts
call    _puts
mov     eax, offset aD ; "%d"
lea     edx, [esp+20h+var_4]
mov     [esp+20h+var_1C], edx
mov     [esp+20h+var_20], eax
call    __isoc99_scanf
mov     edx, [esp+20h+var_4]
mov     eax, offset aYouEnteredD__ ; "You ↴
↳ entered %d... \n"  _printf
mov     [esp+20h+var_1C], edx
mov     [esp+20h+var_20], eax
call    _printf
mov     eax, 0
leave
retn
endp

```

7.1.4 x64

MSVC

Listing 7.1: MSVC 2012 x64

```

_DATA   SEGMENT
$SG1289 DB      'Enter X:', 0aH, 00H
$SG1291 DB      '%d', 00H
$SG1292 DB      'You entered %d...', 0aH, 00H
_DATA  ENDS

_TEXT  SEGMENT
x$ = 32
main  PROC
$LN3:
    sub    rsp, 56
    lea    rcx, OFFSET FLAT:$SG1289 ; 'Enter X:'
    call   printf
    lea    rdx, QWORD PTR x$[rsp]
    lea    rcx, OFFSET FLAT:$SG1291 ; '%d'
    call   scanf
    mov    edx, DWORD PTR x$[rsp]
    lea    rcx, OFFSET FLAT:$SG1292 ; 'You entered %d...'
    call   printf

    ; 0
    xor    eax, eax
    add    rsp, 56
    ret    0
main  ENDP
_TEXT  ENDS

```

GCC

Listing 7.2: Con optimización GCC 4.4.6 x64

```

.LC0:
.string "Enter X:"
.LC1:
.string "%d"

```

```

.LC2:
    .string "You entered %d...\n"

main:
    sub    rsp, 24
    mov    edi, OFFSET FLAT:.LC0 ; "Enter X:"
    call   puts
    lea    rsi, [rsp+12]
    mov    edi, OFFSET FLAT:.LC1 ; "%d"
    xor    eax, eax
    call   __isoc99_scanf
    mov    esi, DWORD PTR [rsp+12]
    mov    edi, OFFSET FLAT:.LC2 ; "You entered %d...\n"
    xor    eax, eax
    call   printf

    ; 0
    xor    eax, eax
    add    rsp, 24
    ret

```

7.1.5 ARM

Con optimización Keil 6/2013 (Modo Thumb)

.text:00000042	scanf_main
.text:00000042	
.text:00000042	var_8 = -8
.text:00000042	
.text:00000042 08 B5	PUSH {R3,LR}
.text:00000044 A9 A0	ADR R0, aEnterX ↴
↳ ; "Enter X:\n"	
.text:00000046 06 F0 D3 F8	BL __2printf
.text:0000004A 69 46	MOV R1, SP
.text:0000004C AA A0	ADR R0, aD ↴
↳ ; "%d"	
.text:0000004E 06 F0 CD F8	BL __0scanf
.text:00000052 00 99	LDR R1, [SP,#8+ ↴
↳ var_8]	
.text:00000054 A9 A0	ADR R0, ↴
↳ aYouEnteredD__ ; "You entered %d...\n"	
.text:00000056 06 F0 CB F8	BL __2printf
.text:0000005A 00 20	MOVS R0, #0
.text:0000005C 08 BD	POP {R3,PC}

ARM64

Listing 7.3: Sin optimización GCC 4.9.1 ARM64

```

1  .LC0:
2      .string "Enter X:"
3  .LC1:
4      .string "%d"
5  .LC2:
6      .string "You entered %d...\n"
7  scanf_main:
8  ; :
9      stp    x29, x30, [sp, -32]!
10 ; (FP=SP)
11     add    x29, sp, 0
12 ;
13     adrp   x0, .LC0
14     add    x0, x0, :lo12:.LC0
15 ; X0=
16 ; :
17     bl     puts
18 ; :
19     adrp   x0, .LC1
20     add    x0, x0, :lo12:.LC1
21 ; (X1=FP+28):
22     add    x1, x29, 28
23 ; X1=
24 ; :
25     bl     __isoc99_scanf
26 ; :
27     ldr    w1, [x29,28]
28 ; W1=x
29 ;
30 ;
31 ; printf()
32     adrp   x0, .LC2
33     add    x0, x0, :lo12:.LC2
34     bl     printf
35 ; 0
36     mov    w0, 0
37 ;
38     ldp    x29, x30, [sp], 32
39     ret

```

7.1.6 MIPS

Listing 7.4: Con optimización GCC 4.4.5 (salida de assembly)

```

$LC0:
    .ascii  "Enter X:\000"
$LC1:
    .ascii  "%d\000"
$LC2:
    .ascii  "You entered %d... \012\000"
main:
; :
    lui      $28,%hi(__gnu_local_gp)
    addiu   $sp,$sp,-40
    addiu   $28,$28,%lo(__gnu_local_gp)
    sw      $31,36($sp)
; puts():
    lw      $25,%call16(puts)($28)
    lui      $4,%hi($LC0)
    jalr    $25
    addiu   $4,$4,%lo($LC0) ; branch delay slot
; scanf():
    lw      $28,16($sp)
    lui      $4,%hi($LC1)
    lw      $25,%call16(__isoc99_scanf)($28)
; scanf(), $a1=$sp+24:
    addiu   $5,$sp,24
    jalr    $25
    addiu   $4,$4,%lo($LC1) ; branch delay slot
; printf():
    lw      $28,16($sp)
; printf(),
; $sp+24:
    lw      $5,24($sp)
    lw      $25,%call16(sprintf)($28)
    lui      $4,%hi($LC2)
    jalr    $25
    addiu   $4,$4,%lo($LC2) ; branch delay slot
; :
    lw      $31,36($sp)
; 0:
    move   $2,$0
; :
    j      $31
    addiu   $sp,$sp,40      ; branch delay slot

```

Listing 7.5: Con optimización GCC 4.4.5 (IDA)

```

.text:00000000 main:
.text:00000000
.text:00000000 var_18          = -0x18
.text:00000000 var_10          = -0x10
.text:00000000 var_4           = -4
.text:00000000
; :
.text:00000000               lui    $gp, (__gnu_local_gp >> 16)
.text:00000004               addiu $sp, -0x28
.text:00000008               la     $gp, (__gnu_local_gp & 0xFF)
.text:0000000C               sw    $ra, 0x28+var_4($sp)
.text:00000010               sw    $gp, 0x28+var_18($sp)
; puts():
.text:00000014               lw    $t9, (puts & 0xFFFF)($gp)
.text:00000018               lui    $a0, ($LC0 >> 16) # "%s"
.text:0000001C               jalr  $t9
.text:00000020               la     $a0, ($LC0 & 0xFFFF) # "%s"
    ↳ "Enter X:" ; branch delay slot
; scanf():
.text:00000024               lw    $gp, 0x28+var_18($sp)
.text:00000028               lui    $a0, ($LC1 >> 16) # "%d"
    ↳ "%d"
.text:0000002C               lw    $t9, (__isoc99_scanf & 0xFFFF)
; scanf(), $a1=$sp+24:
.text:00000030               addiu $a1, $sp, 0x28+var_10
.text:00000034               jalr  $t9 ; branch delay slot
.text:00000038               la     $a0, ($LC1 & 0xFFFF) # "%d"
    ↳ "%d"
; printf():
.text:0000003C               lw    $gp, 0x28+var_18($sp)
; printf(),
; $sp+24:
.text:00000040               lw    $a1, 0x28+var_10($sp)
.text:00000044               lw    $t9, (printf & 0xFFFF)(%s)
    ↳ $gp
.text:00000048               lui    $a0, ($LC2 >> 16) # "%s"
    ↳ "You entered %d...\n"
.text:0000004C               jalr  $t9
.text:00000050               la     $a0, ($LC2 & 0xFFFF) # "%s"
    ↳ "You entered %d...\n" ; branch delay slot

```

```
; :
.text:00000054          lw      $ra, 0x28+var_4($sp)
; 0:
.text:00000058          move   $v0, $zero
; :
.text:0000005C          jr      $ra
.text:00000060          addiu  $sp, 0x28 ; branch delay
    ↳ slot
```

7.2

```
#include <stdio.h>

// 
int x;

int main()
{
    printf ("Enter X:\n");

    scanf ("%d", &x);

    printf ("You entered %d...\n", x);

    return 0;
}
```

7.2.1 MSVC: x86

```
_DATA    SEGMENT
COMM    _x:DWORD
$SG2456  DB     'Enter X:', 0aH, 00H
$SG2457  DB     '%d', 00H
$SG2458  DB     'You entered %d...', 0aH, 00H
_DATA    ENDS
PUBLIC   _main
EXTRN   _scanf:PROC
EXTRN   _printf:PROC
; Function compile flags: /Odtp
_TEXT    SEGMENT
_main    PROC
    push   ebp
    mov    ebp, esp
```

```

push    OFFSET $SG2456
call    _printf
add    esp, 4
push    OFFSET _x
push    OFFSET $SG2457
call    _scanf
add    esp, 8
mov    eax, DWORD PTR _x
push    eax
push    OFFSET $SG2458
call    _printf
add    esp, 8
xor    eax, eax
pop    ebp
ret    0
_main    ENDP
_TEXT    ENDS

```

```
int x=10; //
```

```

_DATA    SEGMENT
_x       DD      0aH
...

```

```

.data:0040FA80 _x          dd ? ; DATA  ↴
    ↳ XREF: _main+10
.data:0040FA80              ; _main ↴
    ↳ +22
.data:0040FA84 dword_40FA84 dd ? ; DATA ↴
    ↳ XREF: _memset+1E
.data:0040FA84              ; ↴
    ↳ unknown_libname_1+28
.data:0040FA88 dword_40FA88 dd ? ; DATA ↴
    ↳ XREF: __sbh_find_block+5
.data:0040FA88              ; ↴
    ↳ __sbh_free_block+2BC
.data:0040FA8C ; LPVOID lpMem
.data:0040FA8C lpMem        dd ? ; DATA ↴
    ↳ XREF: __sbh_find_block+B
.data:0040FA8C              ; ↴
    ↳ __sbh_free_block+2CA
.data:0040FA90 dword_40FA90 dd ? ; DATA ↴
    ↳ XREF: _V6_HeapAlloc+13
.data:0040FA90              ; ↴
    ↳ __calloc_impl+72

```

```
.data:0040FA94 dword_40FA94    dd ?  
↳ XREF: __sbh_free_block+2FE
```

; DATA ↴

7.2.2 MSVC: x86 + OllyDbg

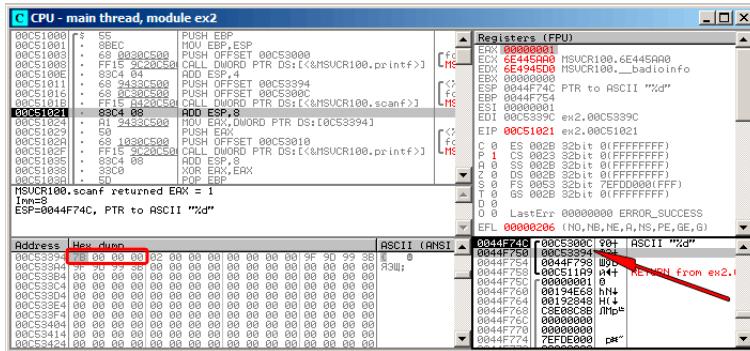


Figura 7.5: OllyDbg:

0x7B.

7B? 00 00 00 7B. endianness, little-endian... : 31 on page 586.

printf().

0x00C53394.

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
0007000000000000	000052000			Heap	Map	RW	R	C:\Windows\System32\locale.nls
0019000000000000	000050000				Priv	RW	RW	
0044C00000000000	000010000			Stack of main thread	Priv	RW	Guar	Guar
0044D00000000000	000030000				Priv	RW	RW	Guar
0075000000000000	000000000			Default heap	Priv	RW	RW	
00C5000000000000	000010000	exe2	.text	PE header	Img	R	E	RWE Cop
00C5100000000000	000010000	exe2	.rdata	Code, imports	Img	R	E	RWE Cop
00C5200000000000	000010000	exe2	.data	Imports	Img	R	E	RWE Cop
00C5300000000000	000010000	exe2	.rsrc	Data	Img	RW	RWE	RWE Cop
00C5400000000000	000010000	exe2	.reloc	Relocations	Img	R	E	RWE Cop
00C5500000000000	000010000	exe2	.text	PE header	Img	R	E	RWE Cop
00E5100000000000	000020000	MSVCRI100	.text	Code, imports, exports	Img	R	E	RWE Cop
00E5100000000000	000020000	MSVCRI100	.data	Data	Img	RW	R	RWE Cop
00E5100000000000	000020000	MSVCRI100	.rsrc	Resources	Img	R	E	RWE Cop
00E5400000000000	000005000	MSVCRI100	.reloc	Relocations	Img	R	E	RWE Cop
00E5500000000000	000010000	MSVCRI100	.text	PE header	Img	R	E	RWE Cop
755D100000000000	000010000		.text	PE header	Img	R	E	RWE Cop
755D500000000000	000010000		.rsrc	Resources	Img	R	E	RWE Cop
755D500000000000	000010000		.reloc	Relocations	Img	R	E	RWE Cop
755E100000000000	000010000	Mod_755E	.text	PE header	Img	R	E	RWE Cop
755E100000000000	000040000	Mod_755E	.data	Data	Img	RW	R	RWE Cop
755E200000000000	000050000	Mod_755E	.rsrc	Resources	Img	R	E	RWE Cop
755E200000000000	000050000	Mod_755E	.reloc	Relocations	Img	R	E	RWE Cop
755E400000000000	000010000	Mod_7564	.text	PE header	Img	R	E	RWE Cop
755E400000000000	000030000	Mod_7564	.data	Data	Img	RW	R	RWE Cop
755E400000000000	000030000	Mod_7564	.rsrc	Resources	Img	R	E	RWE Cop
755E500000000000	000040000	Mod_7564	.reloc	Relocations	Img	R	E	RWE Cop
76F5900000000000	001000000	kernel32	.text	PE header	Img	R	E	RWE Cop
7739000000000000	000010000	kernel32	.data	Code, imports, exports	Img	RW	E	RWE Cop
7739000000000000	000010000	kernel32	.rsrc	Data	Img	R	E	RWE Cop
7739000000000000	000010000	kernel32	.reloc	Resources	Img	R	E	RWE Cop
7721000000000000	000010000	KERNELBASE	.text	PE header	Img	R	E	RWE Cop
7721100000000000	000400000	KERNELBASE	.data	Code, imports, exports	Img	R	E	RWE Cop
7721100000000000	000400000	KERNELBASE	.rsrc	Data	Img	RW	R	RWE Cop
7721100000000000	000400000	KERNELBASE	.reloc	Resources	Img	R	E	RWE Cop
7755400000000000	000030000	KERNELBASE	.text	PE header	Img	R	E	RWE Cop
7755400000000000	000030000	KERNELBASE	.data	Data	Img	RW	R	RWE Cop
7755400000000000	000030000	KERNELBASE	.rsrc	Resources	Img	R	E	RWE Cop
7755400000000000	000030000	KERNELBASE	.reloc	Relocations	Img	R	E	RWE Cop
7755500000000000	000010000	KERNELBASE	.text	PE header	Img	R	E	RWE Cop
7755500000000000	000010000	KERNELBASE	.data	Data	Img	RW	R	RWE Cop
7755500000000000	000010000	KERNELBASE	.rsrc	Resources	Img	R	E	RWE Cop
7755500000000000	000010000	KERNELBASE	.reloc	Relocations	Img	R	E	RWE Cop
7755600000000000	000020000	KERNELBASE	.text	PE header	Img	R	E	RWE Cop
7755600000000000	000020000	KERNELBASE	.data	Data	Img	RW	R	RWE Cop
7755600000000000	000020000	KERNELBASE	.rsrc	Resources	Img	R	E	RWE Cop
7755600000000000	000020000	KERNELBASE	.reloc	Relocations	Img	R	E	RWE Cop
7720000000000000	000010000	ntdll	.text	PE header	Img	R	E	RWE Cop
7721000000000000	000040000	ntdll	.data	Code, exports	Img	RW	R	RWE Cop
7721000000000000	000040000	ntdll	.rsrc	Data	Img	R	E	RWE Cop
7721000000000000	000040000	ntdll	.reloc	Resources	Img	R	E	RWE Cop
7720000000000000	000090000	ntdll	.text	PE header	Img	RW	R	RWE Cop
7720000000000000	000090000	ntdll	.data	Data	Img	RW	R	RWE Cop
7720000000000000	000090000	ntdll	.rsrc	Resources	Img	R	E	RWE Cop
7720000000000000	000090000	ntdll	.reloc	Relocations	Img	R	E	RWE Cop

Figura 7.6: OllyDbg:

7.2.3 GCC: x86

```
; Segment type: Uninitialized  
; Segment permissions: Read/Write
```

```
; Segment type: Pure data  
; Segment permissions: Read/Write
```

7.2.4 MSVC: x64

Listing 7.6: MSVC 2012 x64

```
_DATA SEGMENT
COMM x:DWORD
$SG2924 DB      'Enter X:', 0aH, 00H
$SG2925 DB      '%d', 00H
$SG2926 DB      'You entered %d...', 0aH, 00H
_DATA ENDS

_TEXT SEGMENT
main PROC
$LN3:
```

```

sub    rsp, 40

    lea    rcx, OFFSET FLAT:$SG2924 ; 'Enter X:'
    call   printf
    lea    rdx, OFFSET FLAT:x
    lea    rcx, OFFSET FLAT:$SG2925 ; '%d'
    call   scanf
    mov    edx, DWORD PTR x
    lea    rcx, OFFSET FLAT:$SG2926 ; 'You entered %d...'
    call   printf

    ; 0
    xor    eax, eax

    add    rsp, 40
    ret    0
main  ENDP
_TEXT ENDS

```

x86. .DWORD PTR.

7.2.5 ARM: Con optimización Keil 6/2013 (Modo Thumb)

```

.text:00000000 ; Segment type: Pure code
.text:00000000          AREA .text, CODE
...
.text:00000000 main
.text:00000000          PUSH   {R4,LR}
.text:00000002          ADR    R0, aEnterX     ; "Enter X:
    ↳ X:\n"
.text:00000004          BL     __2printf
.text:00000008          LDR    R1, =x
.text:0000000A          ADR    R0, aD          ; "%d"
.text:0000000C          BL     __0scanf
.text:00000010          LDR    R0, =x
.text:00000012          LDR    R1, [R0]
.text:00000014          ADR    R0, aYouEnteredD__ ; "You entered %d...\n"
    ↳ You entered %d...\n"
.text:00000016          BL     __2printf
.text:0000001A          MOVS   R0, #0
.text:0000001C          POP    {R4,PC}
...
.text:00000020 aEnterX      DCB  "Enter X:",0xA,0     ; DATA
    ↳ XREF: main+2
.text:0000002A          DCB  0
.text:0000002B          DCB  0

```

```

.text:0000002C off_2C          DCD x           ; DATA  ↴
    ↳ XREF: main+8
.text:0000002C                  ; main ↴
    ↳ +10
.text:00000030 aD              DCB "%d",0      ; DATA ↴
    ↳ XREF: main+A
.text:00000033                  DCB 0
.text:00000034 aYouEnteredD___ DCB "You entered %d...",0xA,0 ; ↴
    ↳ DATA XREF: main+14
.text:00000047                  DCB 0
.text:00000047 ; .text         ends
.text:00000047
...
.data:00000048 ; Segment type: Pure data
.data:00000048          AREA .data, DATA
.data:00000048          ; ORG 0x48
.data:00000048          EXPORT x
.data:00000048 x          DCD 0xA           ; DATA ↴
    ↳ XREF: main+8
.data:00000048          ; main ↴
    ↳ +10
.data:00000048 ; .data       ends

```

(*data*).

7.2.6 ARM64

Listing 7.7: Sin optimización GCC 4.9.1 ARM64

```

1
2     .comm   x,4,4
3 .LC0:
4     .string "Enter X:"
5 .LC1:
6     .string "%d"
7 .LC2:
8     .string "You entered %d...\n"
9 f5:
10 ;
11     stp    x29, x30, [sp, -16]!
12 ; (FP=SP)
13     add    x29, sp, 0
14 ;
15     adrp   x0, .LC0
16     add    x0, x0, :lo12:.LC0
17     bl     puts
18 ;

```

```

19      adrp    x0, .LC1
20      add     x0, x0, :lo12:.LC1
21 ; :
22      adrp    x1, x
23      add     x1, x1, :lo12:x
24      bl      __isoc99_scanf
25 ; :
26      adrp    x0, x
27      add     x0, x0, :lo12:x
28 ; :
29      ldr     w1, [x0]
30 ; :
31      adrp    x0, .LC2
32      add     x0, x0, :lo12:.LC2
33      bl      printf
34 ; 0
35      mov     w0, 0
36 ; :
37      ldp     x29, x30, [sp], 16
38      ret

```

7.2.7 MIPS

Listing 7.8: Con optimización GCC 4.4.5 (IDA)

```

.text:004006C0 main:
.text:004006C0
.text:004006C0 var_10          = -0x10
.text:004006C0 var_4           = -4
.text:004006C0
; :
.text:004006C0                 lui     $gp, 0x42
.text:004006C4                 addiu   $sp, -0x20
.text:004006C8                 li      $gp, 0x418940
.text:004006CC                 sw      $ra, 0x20+var_4($sp)
.text:004006D0                 sw      $gp, 0x20+var_10($sp)
; puts():
.text:004006D4                 la      $t9, puts
.text:004006D8                 lui     $a0, 0x40
.text:004006DC                 jalr   $t9 ; puts
.text:004006E0                 la      $a0, aEnterX    # "↙
    ↴ Enter X:" ; branch delay slot
; scanf():
.text:004006E4                 lw      $gp, 0x20+var_10($sp)

```

```

.text:004006E8        lui    $a0, 0x40
.text:004006EC        la     $t9, __isoc99_scanf
; x:
.text:004006F0        la     $a1, x
.text:004006F4        jalr  $t9 ; __isoc99_scanf
.text:004006F8        la     $a0, aD          # "%d" ↵
    ↳ ; branch delay slot
; printf():
.text:004006FC        lw     $gp, 0x20+var_10($sp)
.text:00400700        lui    $a0, 0x40
; x:
.text:00400704        la     $v0, x
.text:00400708        la     $t9, printf
; printf() $a1:
.text:0040070C        lw     $a1, (x - 0x41099C)($v0)
.text:00400710        jalr  $t9 ; printf
.text:00400714        la     $a0, aYouEnteredD__ # ↵
    ↳ "You entered %d...\n" ; branch delay slot
; :
.text:00400718        lw     $ra, 0x20+var_4($sp)
.text:0040071C        move   $v0, $zero
.text:00400720        jr     $ra
.text:00400724        addiu $sp, 0x20 ; branch ↵
    ↳ delay slot

...
.sbss:0041099C # Segment type: Uninitialized
.sbss:0041099C      .sbss
.sbss:0041099C      .globl x
.sbss:0041099C x:   .space 4
.sbss:0041099C

```

Listing 7.9: Con optimización GCC 4.4.5 (objdump)

```

1 004006c0 <main>:
2 ;
3 ;
4 4006c0: 3c1c0042    lui    gp,0x42
5 4006c4: 27bdffe0    addiu $p,sp,-32
6 4006c8: 279c8940    addiu $p,sp,-30400
7 4006cc: afbf001c    sw     ra,28(sp)
8 4006d0: afbc0010    sw     gp,16(sp)
9 ; puts():
10 4006d4: 8f998034   lw     t9,-32716(gp)
11 4006d8: 3c040040   lui    a0,0x40
12 4006dc: 0320f809   jalr  t9

```

```

13    4006e0:    248408f0      addiu   a0,a0,2288 ; branch ↵
     ↳ delay slot
14 ;  scanf():
15    4006e4:    8fbc0010      lw       gp,16(sp)
16    4006e8:    3c040040      lui      a0,0x40
17    4006ec:    8f998038      lw       t9,-32712(gp)
18 ;  x:
19    4006f0:    8f858044      lw       a1,-32700(gp)
20    4006f4:    0320f809      jalr    t9
21    4006f8:    248408fc      addiu   a0,a0,2300 ; branch ↵
     ↳ delay slot
22 ;  printf():
23    4006fc:    8fbc0010      lw       gp,16(sp)
24    400700:    3c040040      lui      a0,0x40
25 ;  x:
26    400704:    8f828044      lw       v0,-32700(gp)
27    400708:    8f99803c      lw       t9,-32708(gp)
28 ;  printf() $a1:
29    40070c:    8c450000      lw       a1,0(v0)
30    400710:    0320f809      jalr    t9
31    400714:    24840900      addiu   a0,a0,2304 ; branch ↵
     ↳ delay slot
32 ;
33    400718:    8fbff001c     lw       ra,28(sp)
34    40071c:    00001021      move    v0,zero
35    400720:    03e00008      jr      ra
36    400724:    27bd0020      addiu   sp,sp,32 ; branch ↵
     ↳ delay slot
37 ;
38    400728:    00200825      move    at,at
39    40072c:    00200825      move    at,at

```

```
int x=10; //
```

Listing 7.10: Con optimización GCC 4.4.5 (IDA)

```

.text:004006A0 main:
.text:004006A0
.text:004006A0 var_10          = -0x10
.text:004006A0 var_8           = -8
.text:004006A0 var_4           = -4

```

```

.text:004006A0          lui      $gp, 0x42
.text:004006A0          addiu   $sp, -0x20
.text:004006A4          li       $gp, 0x418930
.text:004006A8          sw       $ra, 0x20+var_4($sp)
.text:004006AC          sw       $s0, 0x20+var_8($sp)
.text:004006B0          sw       $gp, 0x20+var_10($sp)
.text:004006B4          la       $t9, puts
.text:004006B8          lui      $a0, 0x40
.text:004006BC          jalr    $t9 ; puts
.text:004006C0          la       $a0, aEnterX # "
    ↴ Enter X:"           lw       $gp, 0x20+var_10($sp)
; :
.text:004006CC          lui      $s0, 0x41
.text:004006D0          la       $t9, __isoc99_scanf
.text:004006D4          lui      $a0, 0x40
; :
.text:004006D8          addiu   $a1, $s0, (x - 0x410000)
; $a1.
.text:004006DC          jalr    $t9 ; __isoc99_scanf
.text:004006E0          la       $a0, aD # "%d"
.text:004006E4          lw       $gp, 0x20+var_10($sp)
; :
.text:004006E8          lw       $a1, x
; $a1.
.text:004006EC          la       $t9, printf
.text:004006F0          lui      $a0, 0x40
.text:004006F4          jalr    $t9 ; printf
.text:004006F8          la       $a0, aYouEnteredD__ # "
    ↴ "You entered %d... \n"
.text:004006FC          lw       $ra, 0x20+var_4($sp)
.text:00400700          move   $v0, $zero
.text:00400704          lw       $s0, 0x20+var_8($sp)
.text:00400708          jr       $ra
.text:0040070C          addiu   $sp, 0x20

...
.data:00410920          .globl  x
.data:00410920 x:        .word   0xA

```

Listing 7.11: Con optimización GCC 4.4.5 (objdump)

004006a0 <main>:	
4006a0: 3c1c0042	lui gp, 0x42
4006a4: 27bdffe0	addiu sp, sp, -32

4006a8:	279c8930	addiu	gp, gp, -30416
4006ac:	afbf001c	sw	ra, 28(sp)
4006b0:	afb00018	sw	s0, 24(sp)
4006b4:	afbc0010	sw	gp, 16(sp)
4006b8:	8f998034	lw	t9, -32716(gp)
4006bc:	3c040040	lui	a0, 0x40
4006c0:	0320f809	jalr	t9
4006c4:	248408d0	addiu	a0, a0, 2256
4006c8:	8fb0010	lw	gp, 16(sp)
; :			
4006cc:	3c100041	lui	s0, 0x41
4006d0:	8f998038	lw	t9, -32712(gp)
4006d4:	3c040040	lui	a0, 0x40
; :			
4006d8:	26050920	addiu	a1, s0, 2336
; \$a1.			
4006dc:	0320f809	jalr	t9
4006e0:	248408dc	addiu	a0, a0, 2268
4006e4:	8fb0010	lw	gp, 16(sp)
; \$s0.			
; :			
4006e8:	8e050920	lw	a1, 2336(s0)
; \$a1.			
4006ec:	8f99803c	lw	t9, -32708(gp)
4006f0:	3c040040	lui	a0, 0x40
4006f4:	0320f809	jalr	t9
4006f8:	248408e0	addiu	a0, a0, 2272
4006fc:	8fb001c	lw	ra, 28(sp)
400700:	00001021	move	v0, zero
400704:	8fb00018	lw	s0, 24(sp)
400708:	03e00008	jr	ra
40070c:	27bd0020	addiu	sp, sp, 32

7.3

```
#include <stdio.h>

int main()
{
    int x;
    printf ("Enter X:\n");

    if (scanf ("%d", &x)==1)
        printf ("You entered %d...\n", x);
    else
        printf ("What you entered? Huh?\n");
}
```

```
        return 0;
};
```

scanf()¹

:

```
C:\...>ex3.exe
Enter X:
123
You entered 123...
```

```
C:\...>ex3.exe
Enter X:
ouch
What you entered? Huh?
```

7.3.1 MSVC: x86

(MSVC 2010):

```
    lea      eax, DWORD PTR _x$[ebp]
    push    eax
    push    OFFSET $SG3833 ; '%d', 00H
    call    _scanf
    add    esp, 8
    cmp    eax, 1
    jne    SHORT $LN2@main
    mov    ecx, DWORD PTR _x$[ebp]
    push    ecx
    push    OFFSET $SG3834 ; 'You entered %d...', 0aH, 00H
    call    _printf
    add    esp, 8
    jmp    SHORT $LN1@main
$LN2@main:
    push    OFFSET $SG3836 ; 'What you entered? Huh?', 0aH, 2
    ↓ 00H
    call    _printf
    add    esp, 4
$LN1@main:
    xor    eax, eax
```

¹scanf, wscanf: [MSDN](#)

7.3.2 MSVC: x86: IDA

«exit».

```
.text:00401000 _main proc near
.text:00401000
.text:00401000 var_4 = dword ptr -4
.text:00401000 argc = dword ptr 8
.text:00401000 argv = dword ptr 0Ch
.text:00401000 envp = dword ptr 10h
.text:00401000
.text:00401000     push    ebp
.text:00401001     mov     ebp, esp
.text:00401003     push    ecx
.text:00401004     push    offset Format ; "Enter X:\n"
.text:00401009     call    ds:printf
.text:0040100F     add     esp, 4
.text:00401012     lea     eax, [ebp+var_4]
.text:00401015     push    eax
.text:00401016     push    offset aD ; "%d"
.text:0040101B     call    ds:scanf
.text:00401021     add     esp, 8
.text:00401024     cmp     eax, 1
.text:00401027     jnz    short error
.text:00401029     mov     ecx, [ebp+var_4]
.text:0040102C     push    ecx
.text:0040102D     push    offset aYou ; "You entered %d...\n"
    ↴ "
.text:00401032     call    ds:printf
.text:00401038     add     esp, 8
.text:0040103B     jmp    short exit
.text:0040103D
.text:0040103D error: ; CODE XREF: _main+27
.text:0040103D     push    offset aWhat ; "What you entered? "
    ↴ Huh?\n"
.text:00401042     call    ds:printf
.text:00401048     add     esp, 4
.text:0040104B
.text:0040104B exit: ; CODE XREF: _main+3B
.text:0040104B     xor     eax, eax
.text:0040104D     mov     esp, ebp
.text:0040104F     pop     ebp
.text:00401050     retn
.text:00401050 _main endp
```

```
.  
:  
  
.text:00401000 _text segment para public 'CODE' use32  
.text:00401000      assume cs:_text  
.text:00401000      ;org 401000h  
.text:00401000 ; ask for X  
.text:00401012 ; get X  
.text:00401024      cmp   eax, 1  
.text:00401027      jnz   short error  
.text:00401029 ; print result  
.text:0040103B      jmp   short exit  
.text:0040103D  
.text:0040103D error: ; CODE XREF: _main+27  
.text:0040103D      push offset aWhat ; "What you entered? Huh?  
                   ↴ ?\n"  
.text:00401042      call  ds:printf  
.text:00401048      add   esp, 4  
.text:0040104B exit: ; CODE XREF: _main+3B  
.text:0040104B      xor   eax, eax  
.text:0040104D      mov   esp, ebp  
.text:0040104F      pop   ebp  
.text:00401050      retn  
.text:00401050 _main endp
```

```
; int __cdecl main()
_main proc near

var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h

push    ebp
mov     ebp, esp
push    ecx
push    offset Format ; "Enter X:\n"
call    ds:printf
add    esp, 4
lea     eax, [ebp+var_4]
push    eax
push    offset ad ; "%d"
call    ds:scanf
add    esp, 8
cmp    eax, 1
jnz    short error
```

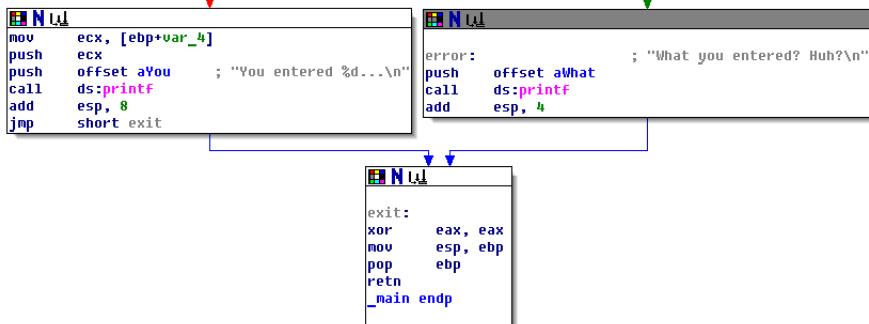


Figura 7.7:

(«group nodes»). :

```
; int __cdecl main()
_main proc near
var_4=dword ptr -4
argc=dword ptr 8
argv=dword ptr 0Ch
envp=dword ptr 10h

push    ebp
mov     ebp, esp
push    ecx
push    offset Format ; "Enter X:\n"
call    ds:printf
add    esp, 4
lea     eax, [ebp+var_4]
push    eax
push    offset ad ; "%d"
call    ds:scanf
add    esp, 8
cmp    eax, 1
jnz    short error
```

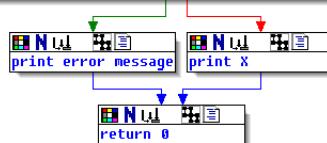


Figura 7.8:

7.3.3 MSVC: x86 + OllyDbg

0x6E494714:

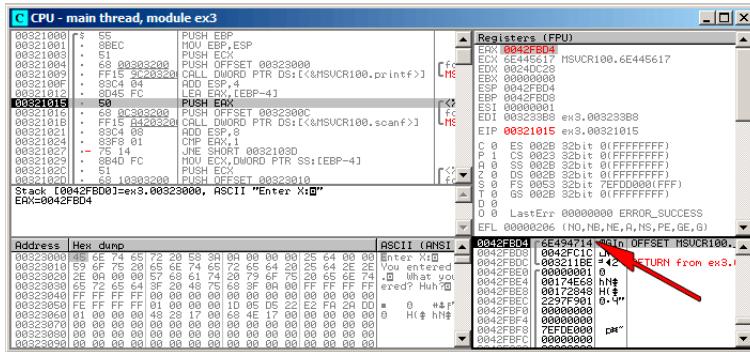


Figura 7.9: OllyDbg: scanf()

`scanf() «asdasd». scanf() EAX, :`

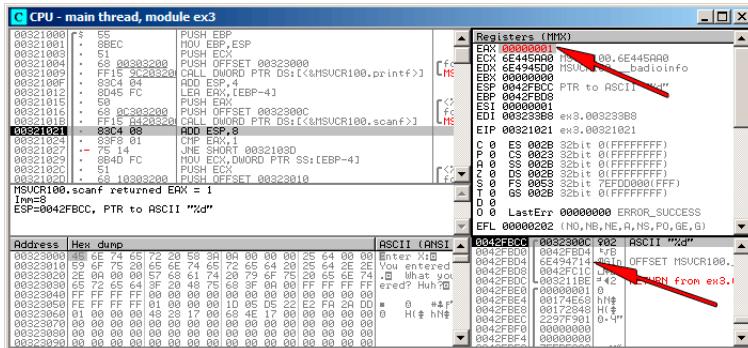


Figura 7.10: OllyDbg: `scanf()`

?.

. EAX, «Set to 1»..

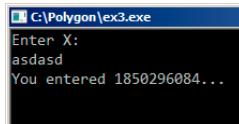


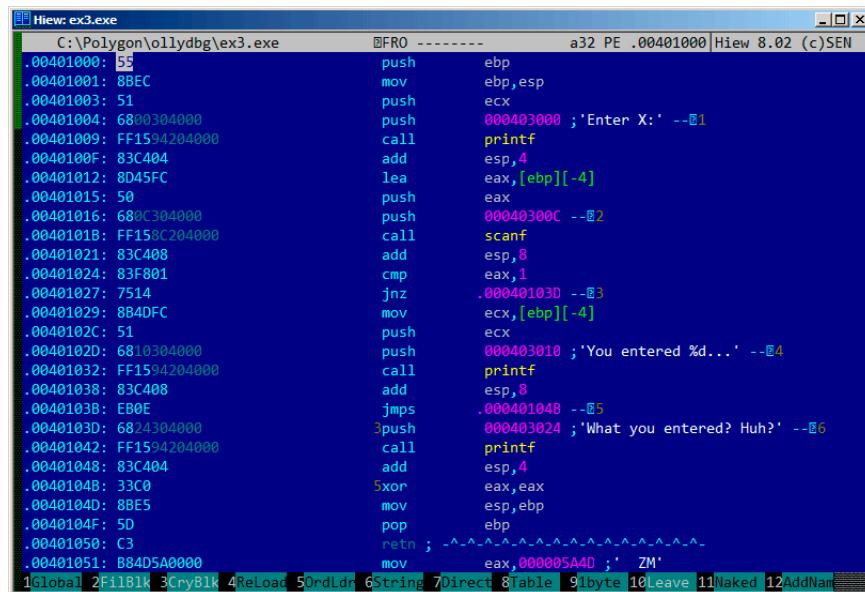
Figura 7.11:

, 1850296084 (0x6E494714)!

7.3.4 MSVC: x86 + Hiew

MSVCR*.DLL (/MD)², main() .text .text (Enter, F8, F6, Enter, Enter).

:



```

Hiew: ex3.exe
C:\Polygon\ollydbg\ex3.exe      DR0 -----  a32 PE .00401000|Hiew 8.02 (c)SEN
.00401000: 55                 push    ebp
.00401001: 8BEC              mov     ebp,esp
.00401003: 51                 push    ecx
.00401004: 6800304000         push    000403000 ;'Enter X:' --@1
.00401009: FF1594204000         call   printf
.0040100F: 83C404              add    esp,4
.00401012: 8D45FC              lea    eax,[ebp][-4]
.00401015: 50                 push    eax
.00401016: 680C304000         push    00040300C --@2
.0040101B: FF158C204000         call   scanf
.00401021: 83C408              add    esp,8
.00401024: 83F801              cmp    eax,1
.00401027: 7514               jnz    .00040103D --@3
.00401029: 8B4DFC              mov    ecx,[ebp][-4]
.0040102C: 51                 push    ecx
.0040102D: 6810304000         push    000403010 ;'You entered %d...' --@4
.00401032: FF1594204000         call   printf
.00401038: 83C408              add    esp,8
.0040103B: EB0E               jmps   .000401048 --@5
.0040103D: 6824304000         push    000403024 ;'What you entered? Huh?' --@6
.00401042: FF1594204000         call   printf
.00401048: 83C404              add    esp,4
.0040104B: 33C0               xor    eax,eax
.0040104D: BEF5               mov    esp,ebp
.0040104F: 5D                 pop    ebp
.00401050: C3                 retn   ; _^_~_~_~_~_~_~_~_~_~_~_~_~_~_~_~_~_~_
.00401051: 8B405A0000         mov    eax,000005A4D ;' ZM'
1Global 2FilBlk 3CryBlk 4ReLoad 50rdLdr 6String 7Direct 8Table 9lbyte 10Leave 11Naked 12AddNm

```

Figura 7.12: Hiew: main()

Hiew ASCII³.

²«dynamic linking»

³ASCII Zero ()

.00401027 (JNZ), F3, «9090»(NOP⁴):

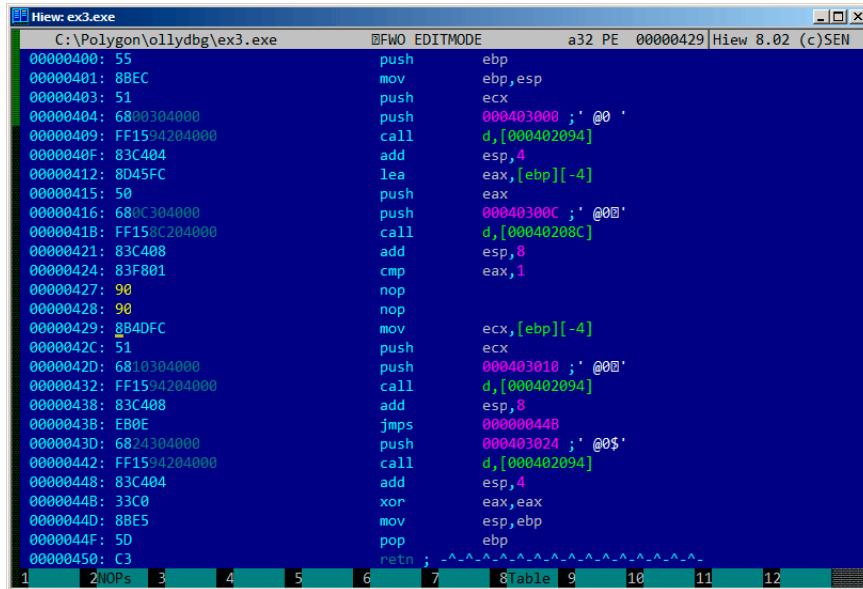


Figura 7.13: Hiew: JNZ NOP

F9 (update).

NOP . JNZ .

7.3.5 MSVC: x64

Listing 7.12: MSVC 2012 x64

```
_DATA SEGMENT
$SG2924 DB      'Enter X:', 0Ah, 00H
$SG2926 DB      '%d', 00H
$SG2927 DB      'You entered %d...', 0Ah, 00H
$SG2929 DB      'What you entered? Huh?', 0Ah, 00H
_DATA ENDS

_TEXT SEGMENT
x$ = 32
```

⁴No OPeration

```

main    PROC
$LN5:
    sub     rsp, 56
    lea     rcx, OFFSET FLAT:$SG2924 ; 'Enter X:'
    call    printf
    lea     rdx, QWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG2926 ; '%d'
    call    scanf
    cmp     eax, 1
    jne     SHORT $LN2@main
    mov     edx, DWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG2927 ; 'You entered %d...'
    call    printf
    jmp     SHORT $LN1@main
$LN2@main:
    lea     rcx, OFFSET FLAT:$SG2929 ; 'What you entered? ↵
    ↳ Huh?'
    call    printf
$LN1@main:
    ; 0
    xor     eax, eax
    add     rsp, 56
    ret     0
main    ENDP
_TEXT   ENDS
END

```

7.3.6 ARM

ARM: Con optimización Keil 6/2013 (Modo Thumb)

Listing 7.13: Con optimización Keil 6/2013 (Modo Thumb)

```

var_8      = -8

        PUSH    {R3,LR}
        ADR    R0, aEnterX      ; "Enter X:\n"
        BL     __2printf
        MOV    R1, SP
        ADR    R0, aD            ; "%d"
        BL     __0scanf
        CMP    R0, #1
        BEQ    loc_1E
        ADR    R0, aWhatYouEntered ; "What you entered? ↵
    ↳ ? Huh?\n"
        BL     __2printf

```

```

loc_1A          MOVS    R0, #0           ; CODE XREF: main+26
                POP     {R3,PC}

loc_1E          LDR     R1, [SP,#8+var_8]   ; CODE XREF: main+12
                ADR     R0, aYouEnteredD__ ; "You entered %d\n"
                ↳ ...\\n"
                BL      __2printf
                B       loc_1A

```

CMP

BEQ⁵ x86.

ARM64

Listing 7.14: Sin optimización GCC 4.9.1 ARM64

```

1 .LC0:
2     .string "Enter X:"
3 .LC1:
4     .string "%d"
5 .LC2:
6     .string "You entered %d...\\n"
7 .LC3:
8     .string "What you entered? Huh?"
9
10 f6:
11 ;
12     stp    x29, x30, [sp, -32]!
13 ; (FP=SP)
14     add    x29, sp, 0
15 ;
16     adrp   x0, .LC0
17     add    x0, x0, :lo12:.LC0
18     bl     puts
19 ;
20     adrp   x0, .LC1
21     add    x0, x0, :lo12:.LC1
22 ;
23     add    x1, x29, 28
24     bl     __isoc99_scanf
25 ; scanf() W0.
26 ;
27     cmp    w0, 1

```

⁵(PowerPC, ARM) Branch if Equal

```

28 ; BNE Branch if Not Equal
29 ; W0<>0,
30         bne      .L2
31 ; W0=1,
32 ;
33         ldr      w1, [x29,28]
34 ; :
35         adrp     x0, .LC2
36         add      x0, x0, :lo12:.LC2
37         bl       printf
38 ; "What you entered? Huh?" :
39         b        .L3
40 .L2:
41 ; "What you entered? Huh?" :
42         adrp     x0, .LC3
43         add      x0, x0, :lo12:.LC3
44         bl       puts
45 .L3:
46 ; 0
47         mov      w0, 0
48 ; :
49         ldp      x29, x30, [sp], 32
50         ret

```

7.3.7 MIPS

Listing 7.15: Con optimización GCC 4.4.5 (IDA)

```

.text:004006A0 main:
.text:004006A0
.text:004006A0 var_18          = -0x18
.text:004006A0 var_10          = -0x10
.text:004006A0 var_4           = -4
.text:004006A0
.text:004006A0                 lui      $gp, 0x42
.text:004006A4                 addiu   $sp, -0x28
.text:004006A8                 li      $gp, 0x418960
.text:004006AC                 sw      $ra, 0x28+var_4($sp)
.text:004006B0                 sw      $gp, 0x28+var_18($sp)
.text:004006B4                 la      $t9, puts
.text:004006B8                 lui      $a0, 0x40
.text:004006BC                 jalr   $t9 ; puts
.text:004006C0                 la      $a0, aEnterX    # "<
             ↴ Enter X:<" 
.text:004006C4                 lw      $gp, 0x28+var_18($sp)
.text:004006C8                 lui      $a0, 0x40
.text:004006CC                 la      $t9, __isoc99_scanf

```

```

.text:004006D0      la    $a0, aD          # "%d"
.text:004006D4      jalr $t9 ; __isoc99_scanf
.text:004006D8      addiu $a1, $sp, 0x28+var_10 # 
    ↳ branch delay slot
.text:004006DC      li    $v1, 1
.text:004006E0      lw    $gp, 0x28+var_18($sp)
.text:004006E4      beq   $v0, $v1, loc_40070C
.text:004006E8      or    $at, $zero      # 
    ↳ branch delay slot, NOP
.text:004006EC      la    $t9, puts
.text:004006F0      lui   $a0, 0x40
.text:004006F4      jalr $t9 ; puts
.text:004006F8      la    $a0, aWhatYouEntered # 
    ↳ "What you entered? Huh?""
.text:004006FC      lw    $ra, 0x28+var_4($sp)
.text:00400700      move $v0, $zero
.text:00400704      jr    $ra
.text:00400708      addiu $sp, 0x28

.text:0040070C loc_40070C:
.text:0040070C      la    $t9, printf
.text:00400710      lw    $a1, 0x28+var_10($sp)
.text:00400714      lui   $a0, 0x40
.text:00400718      jalr $t9 ; printf
.text:0040071C      la    $a0, aYouEnteredD__ # 
    ↳ "You entered %d...\n"
.text:00400720      lw    $ra, 0x28+var_4($sp)
.text:00400724      move $v0, $zero
.text:00400728      jr    $ra
.text:0040072C      addiu $sp, 0x28

```

BEQ «Branch Equal».

7.3.8 Ejercicio

7.4 Ejercicios

7.4.1 Ejercicio #1

```

#include <string.h>
#include <stdio.h>

void alter_string(char *s)
{
    strcpy (s, "Goodbye!");
    printf ("Result: %s\n", s);

```

```
};  
  
int main()  
{  
    alter_string ("Hello, world!\n");  
}
```

Responda: [G.1.3 on page 1260](#).

Capítulo 8

Listing 8.1:

```
#include <stdio.h>

int f (int a, int b, int c)
{
    return a*b+c;
}

int main()
{
    printf ("%d\n", f(1, 2, 3));
    return 0;
}
```

8.1 x86

8.1.1 MSVC

(MSVC 2010 Express):

Listing 8.2: MSVC 2010 Express

```
_TEXT  SEGMENT
_a$ = 8
    ↓ = 4
; size ↴
_b$ = 12
    ↓ = 4
; size ↴
_c$ = 16
    ↓ = 4
; size ↴
_f      PROC
```

```

    push    ebp
    mov     ebp,  esp
    mov     eax, DWORD PTR _a$[ebp]
    imul   eax, DWORD PTR _b$[ebp]
    add    eax, DWORD PTR _c$[ebp]
    pop    ebp
    ret    0
_f      ENDP

_main  PROC
    push    ebp
    mov     ebp,  esp
    push   3 ;
    push   2 ;
    push   1 ;
    call   _f
    add    esp, 12
    push   eax
    push   OFFSET $SG2463 ; '%d', 0aH, 00H
    call   _printf
    add    esp, 8
    ; 0
    xor    eax, eax
    pop    ebp
    ret    0
_main  ENDP

```

8.1.2 MSVC + OllyDbg

OllyDbg.

OllyDbg «RETURN from» y «Arg1 = ...», Spanish text placeholder.

N.B.:

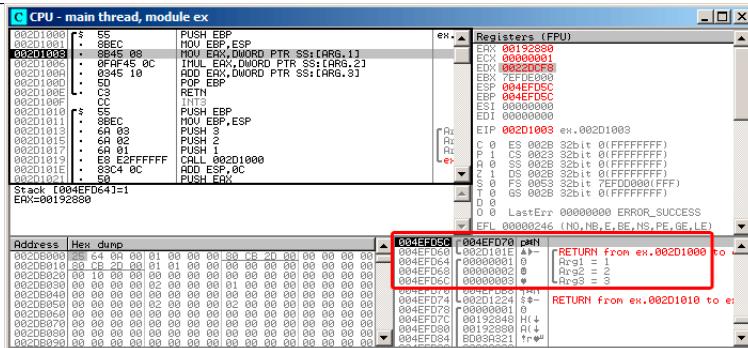


Figura 8.1: OllyDbg: f()

8.1.3 GCC

Listing 8.3: GCC 4.4.1

```

public f
f
proc near

arg_0      = dword ptr  8
arg_4      = dword ptr 0Ch
arg_8      = dword ptr 10h

push    ebp
mov     ebp, esp
mov     eax, [ebp+arg_0] ;
imul   eax, [ebp+arg_4] ;
add    eax, [ebp+arg_8] ;
pop    ebp
retn
f
endp

public main
main
proc near

var_10     = dword ptr -10h
var_C      = dword ptr -0Ch
var_8      = dword ptr -8

push    ebp
mov     ebp, esp
and    esp, 0FFFFFFF0h
sub    esp, 10h

```

```

        mov    [esp+10h+var_8], 3 ;
        mov    [esp+10h+var_C], 2 ;
        mov    [esp+10h+var_10], 1 ;
        call   f
        mov    edx, offset aD ; "%d\n"
        mov    [esp+10h+var_C], eax
        mov    [esp+10h+var_10], edx
        call   _printf
        mov    eax, 0
        leave
        retn
main  endp

```

LEAVE ([A.6.2 on page 1233](#))

8.2 x64

8.2.1 MSVC

Con optimización MSVC:

Listing 8.4: Con optimización MSVC 2012 x64

```

$SG2997 DB      '%d', 0aH, 00H

main  PROC
        sub    rsp, 40
        mov    edx, 2
        lea    r8d, QWORD PTR [rdx+1] ; R8D=3
        lea    ecx, QWORD PTR [rdx-1] ; ECX=1
        call   f
        lea    rcx, OFFSET FLAT:$SG2997 ; '%d'
        mov    edx, eax
        call   printf
        xor    eax, eax
        add    rsp, 40
        ret    0
main  ENDP

f     PROC
; ECX -
; EDX -
; R8D -
        imul   ecx, edx
        lea    eax, DWORD PTR [r8+rcx]
        ret    0

```

f	ENDP
---	------

:

Listing 8.5: MSVC 2012 x64

```

f          proc near
; shadow space:
arg_0      = dword ptr  8
arg_8      = dword ptr  10h
arg_10     = dword ptr  18h

; ECX -
; EDX -
; R8D -
mov        [rsp+arg_10], r8d
mov        [rsp+arg_8], edx
mov        [rsp+arg_0], ecx
mov        eax, [rsp+arg_0]
imul      eax, [rsp+arg_8]
add       eax, [rsp+arg_10]
retn
f          endp

main      proc near
sub       rsp, 28h
mov       r8d, 3 ;
mov       edx, 2 ;
mov       ecx, 1 ;
call      f
mov       edx, eax
lea       rcx, $SG2931    ; "%d\n"
call      printf

; 0
xor       eax, eax
add       rsp, 28h
retn
main      endp

```

«shadow space»¹: 1) ² 2) ².

¹MSDN

²MSDN

8.2.2 GCC

Con optimización GCC :

Listing 8.6: Con optimización GCC 4.4.6 x64

```
f:
; EDI -
; ESI -
; EDX -
imul    esi, edi
lea     eax, [rdx+rsi]
ret

main:
sub    rsp, 8
mov   edx, 3
mov   esi, 2
mov   edi, 1
call  f
mov   edi, OFFSET FLAT:.LC0 ; "%d\n"
mov   esi, eax
xor   eax, eax ;
call  printf
xor   eax, eax
add   rsp, 8
ret
```

Sin optimización GCC:

Listing 8.7: GCC 4.4.6 x64

```
f:
; EDI -
; ESI -
; EDX -
push   rbp
mov    rbp, rsp
mov    DWORD PTR [rbp-4], edi
mov    DWORD PTR [rbp-8], esi
mov    DWORD PTR [rbp-12], edx
mov    eax, DWORD PTR [rbp-4]
imul   eax, DWORD PTR [rbp-8]
add    eax, DWORD PTR [rbp-12]
leave
ret

main:
```

```

push    rbp
mov     rbp, rsp
mov     edx, 3
mov     esi, 2
mov     edi, 1
call    f
mov     edx, eax
mov     eax, OFFSET FLAT:.LC0 ; "%d\n"
mov     esi, edx
mov     rdi, rax
mov     eax, 0 ;
call    printf
mov     eax, 0
leave
ret

```

8.2.3 GCC: uint64_t int

:

```

#include <stdio.h>
#include <stdint.h>

uint64_t f (uint64_t a, uint64_t b, uint64_t c)
{
    return a*b+c;
}

int main()
{
    printf ("%lld\n", f(0x1122334455667788,
                        0x1111111222222222,
                        0x3333333444444444));
    return 0;
}

```

Listing 8.8: Con optimización GCC 4.4.6 x64

```

f          proc near
          imul    rsi, rdi
          lea     rax, [rdx+rsi]
          retn
f          endp

main      proc near
          sub    rsp, 8

```

```

    mov    rdx, 333333344444444h ;
    mov    rsi, 11111112222222h ;
    mov    rdi, 1122334455667788h ;
    call   f
    mov    edi, offset format ; "%lld\n"
    mov    rsi, rax
    xor    eax, eax ;
    call   _printf
    xor    eax, eax
    add    rsp, 8
    retn
main
endp

```

8.3 ARM

8.3.1 Sin optimización Keil 6/2013 (Modo ARM)

.text:000000A4 00 30 A0 E1	MOV	R3, R0
.text:000000A8 93 21 20 E0	MLA	R0, R3, R1, ↴ R2
.text:000000AC 1E FF 2F E1	BX	LR
...		
.text:000000B0 main		
.text:000000B0 10 40 2D E9	STMFD	SP!, {R4,LR}
.text:000000B4 03 20 A0 E3	MOV	R2, #3
.text:000000B8 02 10 A0 E3	MOV	R1, #2
.text:000000BC 01 00 A0 E3	MOV	R0, #1
.text:000000C0 F7 FF FF EB	BL	f
.text:000000C4 00 40 A0 E1	MOV	R4, R0
.text:000000C8 04 10 A0 E1	MOV	R1, R4
.text:000000CC 5A 0F 8F E2	ADR	R0, aD_0 ↴ ; "%d\n"
.text:000000D0 E3 18 00 EB	BL	_2printf
.text:000000D4 00 00 A0 E3	MOV	R0, #0
.text:000000D8 10 80 BD E8	LDMFD	SP!, {R4,PC}

f()

MLA (Multiply Accumulate)

³ (*Fused multiply-add*) ⁴.

MOV R3, R0,

³ Spanish text placeholder

⁴ wikipedia

8.3.2 Con optimización Keil 6/2013 (Modo ARM)

```
.text:00000098          f
.text:00000098 91 20 20 E0           MLA      R0, R1, R0, 
    ↴ R2
.text:0000009C 1E FF 2F E1           BX       LR
```

(-03).

8.3.3 Con optimización Keil 6/2013 (Modo Thumb)

```
.text:0000005E 48 43           MULS    R0, R1
.text:00000060 80 18           ADDS    R0, R0, R2
.text:00000062 70 47           BX      LR
```

8.3.4 ARM64

Con optimización GCC (Linaro) 4.9

Listing 8.9: Con optimización GCC (Linaro) 4.9

```
f:
    madd    w0, w0, w1, w2
    ret

main:
; :
    stp     x29, x30, [sp, -16]!
    mov     w2, 3
    mov     w1, 2
    add     x29, sp, 0
    mov     w0, 1
    bl      f
    mov     w1, w0
    adrp   x0, .LC7
    add     x0, x0, :lo12:.LC7
    bl      printf
; 0
    mov     w0, 0
;
    ldp     x29, x30, [sp], 16
    ret
```

```
.LC7:
    .string "%d\n"
```

```
#include <stdio.h>
#include <stdint.h>

uint64_t f (uint64_t a, uint64_t b, uint64_t c)
{
    return a*b+c;
}

int main()
{
    printf ("%lld\n", f(0x1122334455667788,
                        0x1111111222222222,
                        0x3333333444444444));
    return 0;
}
```

```
f:
    madd    x0, x0, x1, x2
    ret
main:
    mov     x1, 13396
    adrp   x0, .LC8
    stp    x29, x30, [sp, -16]!
    movk   x1, 0x27d0, lsl 16
    add    x0, x0, :lo12:.LC8
    movk   x1, 0x122, lsl 32
    add    x29, sp, 0
    movk   x1, 0x58be, lsl 48
    bl     printf
    mov    w0, 0
    ldp    x29, x30, [sp], 16
    ret

.LC8:
    .string "%lld\n"
```

: 28.3.1 on page 575.

Sin optimización GCC (Linaro) 4.9

```
f:
    sub    sp, sp, #16
    str    w0, [sp,12]
```

```

str    w1, [sp,8]
str    w2, [sp,4]
ldr    w1, [sp,12]
ldr    w0, [sp,8]
mul   w1, w1, w0
ldr    w0, [sp,4]
add   w0, w1, w0
add   sp, sp, 16
ret

```

Register Save Area. [ARM13c] «Shadow Space»: 8.2.1 on page 118.

8.4 MIPS

Listing 8.10: Con optimización GCC 4.4.5

```

.text:00000000 f:
; $a0=a
; $a1=b
; $a2=c
.text:00000000          mult    $a1, $a0
.text:00000004          mflo    $v0
.text:00000008          jr      $ra
.text:0000000C          addu    $v0, $a2, $v0      ; ↴
                           ↴ branch delay slot
;

.text:00000010 main:
.text:00000010
.text:00000010 var_10      = -0x10
.text:00000010 var_4       = -4
.text:00000010
.text:00000010          lui     $gp, (__gnu_local_gp >> ↴
                           ↴ 16)
.text:00000014          addiu   $sp, -0x20
.text:00000018          la      $gp, (__gnu_local_gp & 0x
                           ↴ xFFF)
.text:0000001C          sw     $ra, 0x20+var_4($sp)
.text:00000020          sw     $gp, 0x20+var_10($sp)
; c:
.text:00000024          li     $a2, 3
; a:
.text:00000028          li     $a0, 1
.text:0000002C          jal    f
; b:

```

```

.text:00000030          li      $a1, 2           ; ↴
    ↳ branch delay slot
;
.text:00000034          lw      $gp, 0x20+var_10($sp)
.text:00000038          lui      $a0, ($LC0 >> 16)
.text:0000003C          lw      $t9, (printf & 0xFFFF)(↖
    ↳ $gp)
.text:00000040          la      $a0, ($LC0 & 0xFFFF)
.text:00000044          jalr    $t9
;
.text:00000048          move    $a1, $v0          ; ↴
    ↳ branch delay slot
.text:0000004C          lw      $ra, 0x20+var_4($sp)
.text:00000050          move    $v0, $zero
.text:00000054          jr      $ra
.text:00000058          addiu   $sp, 0x20          ; ↴
    ↳ branch delay slot

```

ADD y ADDU. ADDU .

\$V0.

JAL («Jump and Link»).

Capítulo 9

[1](#)

9.1

```
push envp  
push argv  
push argc  
call main  
push eax  
call exit
```

```
exit(main(argc,argv,envp));
```

Lo más probable es que quede un valor aleatorio de la ejecución de tu función, así que el código de salida será seudorandom.

. `main()` void:

```
#include <stdio.h>  
  
void main()  
{  
    printf ("Hello, world!\n");  
}
```

Linux.

GCC 4.8.1 `printf()` `puts()` ([3.4.3 on page 21](#)) , `puts()` `printf()`. EAX `main()`.

¹Spanish text placeholder: MSDN: Return Values (C++): [MSDN](#)

Listing 9.1: GCC 4.8.1

```
.LC0:
    .string "Hello, world!"
main:
    push    ebp
    mov     ebp, esp
    and     esp, -16
    sub     esp, 16
    mov     DWORD PTR [esp], OFFSET FLAT:.LC0
    call    puts
    leave
    ret
```

:

Listing 9.2: tst.sh

```
#!/bin/sh
./hello_world
echo $?
```

:

```
$ tst.sh
Hello, world!
14
```

14 .

9.2

```
int f()
{
    // skip first 3 random values
    rand();
    rand();
    rand();
    // and use 4th
    return rand();
}
```

9.3

```

struct s
{
    int a;
    int b;
    int c;
};

struct s get_some_values (int a)
{
    struct s rt;

    rt.a=a+1;
    rt.b=a+2;
    rt.c=a+3;

    return rt;
}

```

... (MSVC 2010 /0x):

```

$T3853 = 8          ; size = 4
_a$ = 12           ; size = 4
?get_some_values@@YA?AUt@@H@Z PROC      ; get_some_values
    mov    ecx, DWORD PTR _a$[esp-4]
    mov    eax, DWORD PTR $T3853[esp-4]
    lea    edx, DWORD PTR [ecx+1]
    mov    DWORD PTR [eax], edx
    lea    edx, DWORD PTR [ecx+2]
    add    ecx, 3
    mov    DWORD PTR [eax+4], edx
    mov    DWORD PTR [eax+8], ecx
    ret    0
?get_some_values@@YA?AUt@@H@Z ENDP      ; get_some_values

```

:

```

struct s
{
    int a;
    int b;
    int c;
};

struct s get_some_values (int a)
{
    return (struct s){.a=a+1, .b=a+2, .c=a+3};
}

```

Listing 9.3: GCC 4.8.1

```
_get_some_values proc near

ptr_to_struct    = dword ptr  4
a                = dword ptr  8

        mov     edx,  [esp+a]
        mov     eax,  [esp+ptr_to_struct]
        lea     ecx,  [edx+1]
        mov     [eax], ecx
        lea     ecx,  [edx+2]
        add     edx,  3
        mov     [eax+4], ecx
        mov     [eax+8], edx
        retn

_get_some_values endp
```

Capítulo 10

10.1

```
#include <stdio.h>

void f1 (int x, int y, int *sum, int *product)
{
    *sum=x+y;
    *product=x*y;
};

int sum, product;

void main()
{
    f1(123, 456, &sum, &product);
    printf ("sum=%d, product=%d\n", sum, product);
}
```

:

Listing 10.1: Con optimización MSVC 2010 (/Ob0)

```
COMM      _product:DWORD
COMM      _sum:DWORD
$SG2803 DB      'sum=%d, product=%d', 0aH, 00H

_x$ = 8                                     ; size 2
    ↓ = 4
_y$ = 12                                    ; size 2
    ↓ = 4
_sum$ = 16                                   ; size 2
    ↓ = 4
```

```

_product$ = 20 ; size ↴
    ↳ = 4
_f1    PROC
        mov     ecx, DWORD PTR _y$[esp-4]
        mov     eax, DWORD PTR _x$[esp-4]
        lea     edx, DWORD PTR [eax+ecx]
        imul   eax, ecx
        mov     ecx, DWORD PTR _product$[esp-4]
        push   esi
        mov     esi, DWORD PTR _sum$[esp]
        mov     DWORD PTR [esi], edx
        mov     DWORD PTR [ecx], eax
        pop    esi
        ret    0
_f1    ENDP

_main  PROC
        push   OFFSET _product
        push   OFFSET _sum
        push   456 ; ↴
    ↳ 0000001c8H
        push   123 ; ↴
    ↳ 00000007bH
        call   _f1
        mov    eax, DWORD PTR _product
        mov    ecx, DWORD PTR _sum
        push   eax
        push   ecx
        push   OFFSET $SG2803
        call   DWORD PTR __imp__printf
        add    esp, 28 ; ↴
    ↳ 00000001cH
        xor    eax, eax
        ret    0
_main  ENDP

```

OllyDbg:

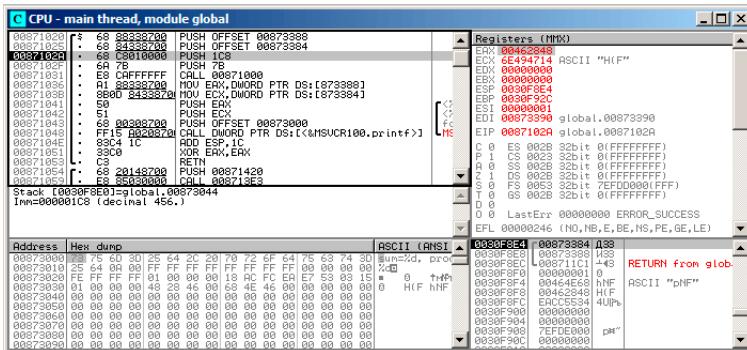


Figura 10.1: OllyDbg: f1()

f1(). «Follow in dump»

Alt-M :

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00050000	00004000			Stack of main thread	Map	R		
00051000	00001000				Priv	R/W		
00072000	00007000				Map	R		
00152000	00007000				Priv	R/W		C:\Windows\System32\loc...
00301000	00001000				Map	R/W		
00302000	00001000				Priv	R/W		
00460000	00005000				Priv	R/W		
00440000	00007000				Priv	R/W		
00501000	00001000				Priv	R/W		
00570000	00001000	global		Default heap	Img	R		R/W E Cop
00571000	00001000	global	.text	PE header	Img	R E		R/W E Cop
00572000	00001000	global	.idata	Code	Img	R		R/W E Cop
00573000	00001000	global	.data	Imports	Img	R		R/W E Cop
00574000	00001000	global	.rsrc	Data	Img	R		R/W E Cop
00574400	00001000	global	.reloc	Relocations	Img	R		R/W E Cop
6E3E1000	00001000			PE header	Img	R		R/W E Cop
6E3E2000	00002000	MSVCRI00	.text	Code, imports, exports	Img	R E		R/W E Cop
6E493000	00006000	MSVCRI00	.data	Data	Img	R		R/W E Cop
6E494000	00001000	MSVCRI00	.rsrc	Resources	Img	R		R/W E Cop
6E495000	00001000	MSVCRI00	.reloc	Relocations	Img	R		R/W E Cop
755D0000	00001000	Mod_7550		PE header	Img	R		R/W E Cop
755D1000	00003000	Mod_7550			Img	R E		R/W E Cop
755D2000	00001000	Mod_7550			Img	R		R/W E Cop
755D3000	00003000	Mod_7550			Img	R		R/W E Cop
755E0000	00001000	Mod_755E		PE header	Img	R		R/W E Cop
755E1000	00001000	Mod_755E			Img	R E		R/W E Cop
755E2000	00005000	Mod_755E			Img	R		R/W E Cop
75630000	00009000				Img	R		R/W E Cop

Figura 10.2: OllyDbg:

(F7) f1():

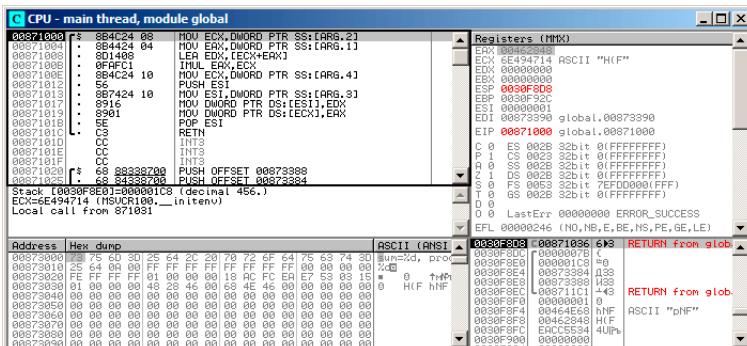


Figura 10.3: OllyDbg:

456 (0x1C8) y 123 (0x7B), .

f1():

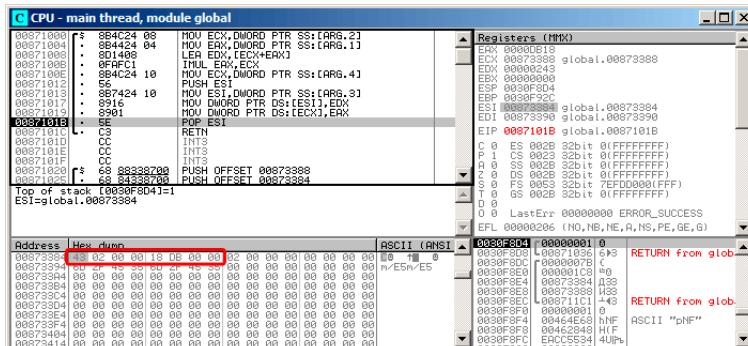


Figura 10.4: OllyDbg: f1()

printf():

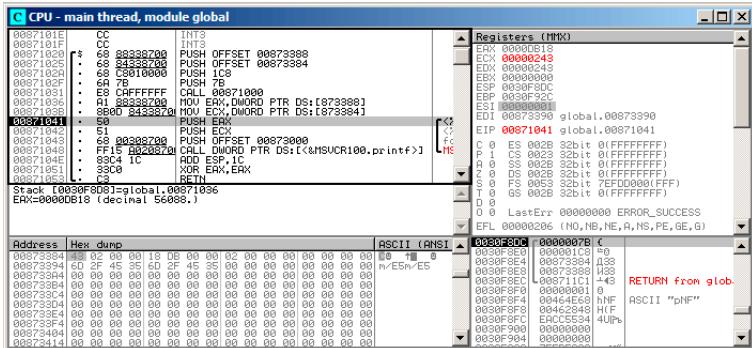


Figura 10.5: OllyDbg: printf()

10.2

1

Listing 10.2:

```
void main()
{
    int sum, product; //

    f1(123, 456, &sum, &product);
    printf ("sum=%d, product=%d\n", sum, product);
};
```

f1(), i:

Listing 10.3: Con optimización MSVC 2010 (/Ob0)

```
_product$ = -8
    ↳ = 4
_sum$ = -4
    ↳ = 4
_main    PROC
; Line 10
        sub    esp, 8
; Line 13
        lea    eax, DWORD PTR _product$[esp+8]
        push   eax
        lea    ecx, DWORD PTR sum$[esp+12]
```

```
    push    ecx
    push    456
    ↳ 000001c8H ; ↴
    push    123
    ↳ 0000007bH ; ↴
    call    _f1
; Line 14
    mov     edx, DWORD PTR _product$[esp+24]
    mov     eax, DWORD PTR _sum$[esp+24]
    push    edx
    push    eax
    push    OFFSET $SG2803
    call    DWORD PTR __imp__printf
; Line 15
    xor     eax, eax
    add     esp, 36 ; ↴
    ↳ 00000024H
    ret    0
```

OllyDbg. 0x2EF854 y 0x2EF855. :

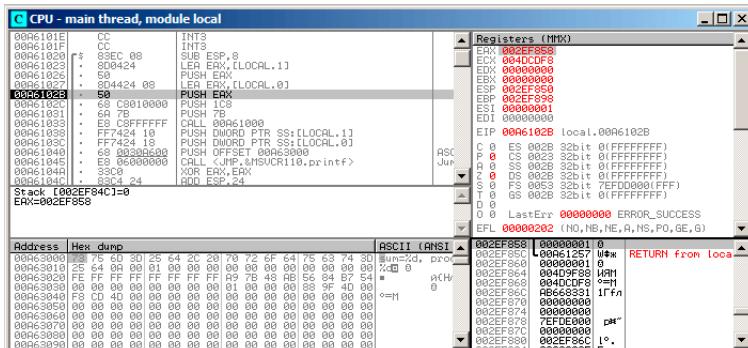


Figura 10.6: OllyDbg:

. 0x2EF854 y 0x2EF858:

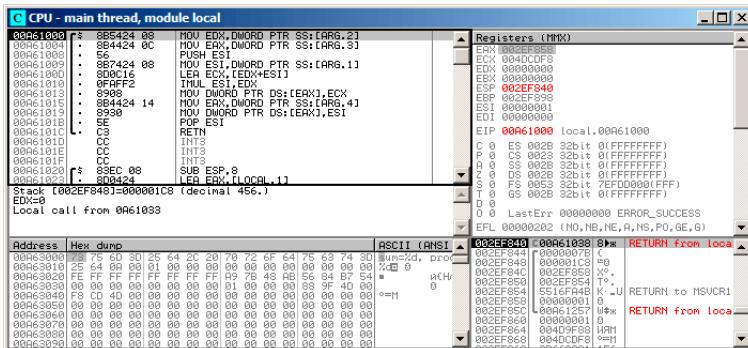


Figura 10.7: OllyDbg: f1()

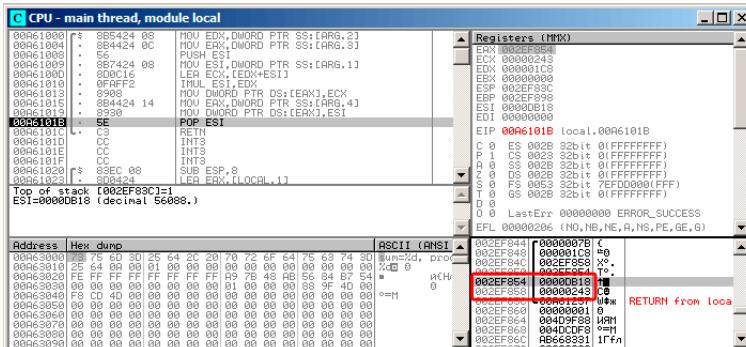


Figura 10.8: OllyDbg: f1()

10.3 Conclusión

references . : (51.3 on page 723).

Capítulo 11

[Dij68], [Knu74], [Yur13, pág. 1.3.2].

:

```
#include <stdio.h>

int main()
{
    printf ("begin\n");
    goto exit;
    printf ("skip me!\n");
exit:
    printf ("end\n");
}
```

MSVC 2012:

Listing 11.1: MSVC 2012

```
$SG2934 DB      'begin', 0aH, 00H
$SG2936 DB      'skip me!', 0aH, 00H
$SG2937 DB      'end', 0aH, 00H

_main PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG2934 ; 'begin'
    call    _printf
    add    esp, 4
    jmp    SHORT $exit$3
    push    OFFSET $SG2936 ; 'skip me!'
    call    _printf
    add    esp, 4
$exit$3:
```

```
push    OFFSET $SG2937 ; 'end'
call    _printf
add    esp, 4
xor    eax, eax
pop    ebp
ret    0
_main  ENDP
```

La segunda llamada a `printf()` solo podría ejecutarse con intervención humana, usando un depurador o parcheando el código.

Hiew:

Figura 11.1: Hiew

Figura 11.2: Hiew

```
C:\Polygon>goto.exe  
  
begin  
skip me!  
end
```

Figura 11.3:

11.1

Listing 11.2: Con optimización MSVC 2012

```
$SG2981 DB      'begin', 0aH, 00H
$SG2983 DB      'skip me!', 0aH, 00H
$SG2984 DB      'end', 0aH, 00H

_main    PROC
        push    OFFSET $SG2981 ; 'begin'
        call    _printf
        push    OFFSET $SG2984 ; 'end'
$exit$4:
        call    _printf
        add     esp, 8
        xor     eax, eax
        ret
_main    ENDP
```

11.2 Ejercicio

Capítulo 12

12.1

```
#include <stdio.h>

void f_signed (int a, int b)
{
    if (a>b)
        printf ("a>b\n");
    if (a==b)
        printf ("a==b\n");
    if (a<b)
        printf ("a<b\n");
};

void f_unsigned (unsigned int a, unsigned int b)
{
    if (a>b)
        printf ("a>b\n");
    if (a==b)
        printf ("a==b\n");
    if (a<b)
        printf ("a<b\n");
};

int main()
{
    f_signed(1, 2);
    f_unsigned(1, 2);
    return 0;
};
```

12.1.1 x86

x86 + MSVC

Listing 12.1: Sin optimización MSVC 2010

```

_a$ = 8
_b$ = 12
_f_signed PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    cmp     eax, DWORD PTR _b$[ebp]
    jle     SHORT $LN3@f_signed
    push    OFFSET $SG737           ; 'a>b'
    call    _printf
    add     esp, 4
$LN3@f_signed:
    mov     ecx, DWORD PTR _a$[ebp]
    cmp     ecx, DWORD PTR _b$[ebp]
    jne     SHORT $LN2@f_signed
    push    OFFSET $SG739           ; 'a==b'
    call    _printf
    add     esp, 4
$LN2@f_signed:
    mov     edx, DWORD PTR _a$[ebp]
    cmp     edx, DWORD PTR _b$[ebp]
    jge     SHORT $LN4@f_signed
    push    OFFSET $SG741           ; 'a<b'
    call    _printf
    add     esp, 4
$LN4@f_signed:
    pop    ebp
    ret    0
_f_signed ENDP

```

Jump if Less or Equal. JNE: *Jump if Not Equal.* .

JGE: *Jump if Greater or Equal.*

f_unsigned()

Listing 12.2: GCC

```

_a$ = 8    ; size = 4
_b$ = 12   ; size = 4
_f_unsigned PROC
    push    ebp
    mov     ebp, esp

```

```

    mov    eax, DWORD PTR _a$[ebp]
    cmp    eax, DWORD PTR _b$[ebp]
    jbe    SHORT $LN3@f_unsigned
    push   OFFSET $SG2761      ; 'a>b'
    call   _printf
    add    esp, 4
$LN3@f_unsigned:
    mov    ecx, DWORD PTR _a$[ebp]
    cmp    ecx, DWORD PTR _b$[ebp]
    jne    SHORT $LN2@f_unsigned
    push   OFFSET $SG2763      ; 'a==b'
    call   _printf
    add    esp, 4
$LN2@f_unsigned:
    mov    edx, DWORD PTR _a$[ebp]
    cmp    edx, DWORD PTR _b$[ebp]
    jae    SHORT $LN4@f_unsigned
    push   OFFSET $SG2765      ; 'a<b'
    call   _printf
    add    esp, 4
$LN4@f_unsigned:
    pop    ebp
    ret    0
_f_unsigned ENDP

```

JBEJump if Below or Equal y JAEJump if Above or Equal. (JA/JAE/JB/JBE) JG/JGE/JL/JLE

(30 on page 584).

Listing 12.3: main()

```

_main PROC
    push   ebp
    mov    ebp, esp
    push   2
    push   1
    call   _f_signed
    add    esp, 8
    push   2
    push   1
    call   _f_unsigned
    add    esp, 8
    xor    eax, eax
    pop    ebp
    ret    0
_main ENDP

```

x86 + MSVC + OllyDbg

. f_unsigned()

2

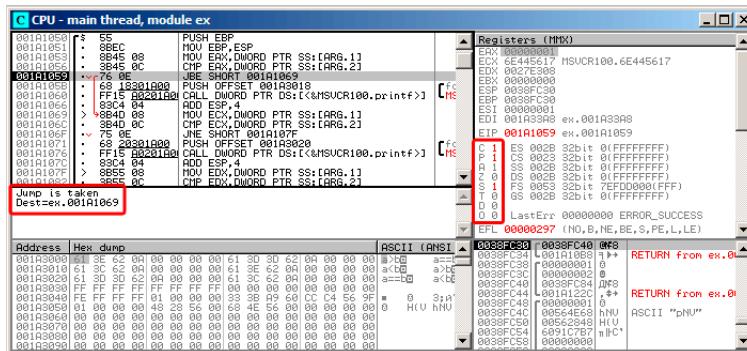


Figura 12.1: OllyDbg: f_unsigned():

: C=1, P=1, A=1, Z=0, S=1, T=0, D=0, O=0.

OllyDbg (JBE) . [Int13], JBE CF=1 o ZF=1. .

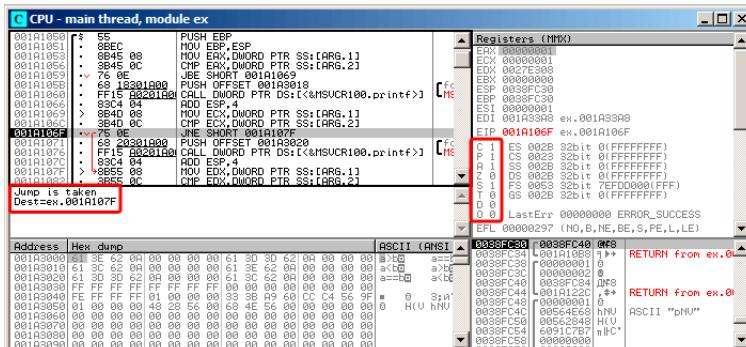


Figura 12.2: OllyDbg: f_unsigned():

OllyDbg JNZ. , JNZ ZF=0 (zero flag).

JNB:

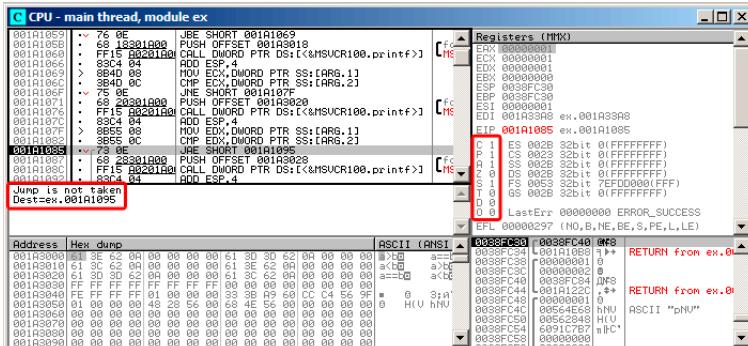


Figura 12.3: OllyDbg: f_unsigned():

[Int13] JNB CF=0 (carry flag). printf().

f_signed().

: C=1, P=1, A=1, Z=0, S=1, T=0, D=0, O=0.

JLE:

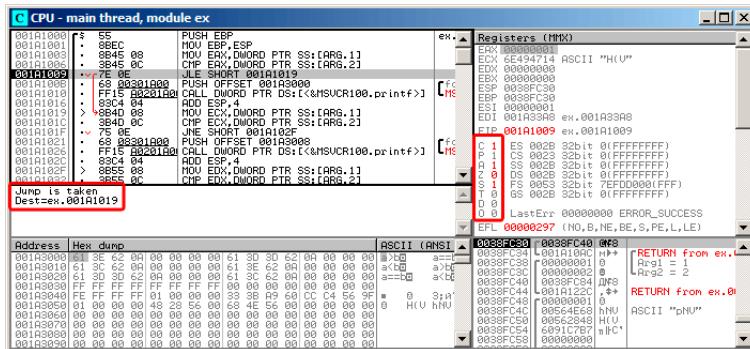


Figura 12.4: OllyDbg: f_signed():

[Int13] ZF=1 o SF≠OF. SF≠OF.

JNZ: ZF=0 (zero flag):

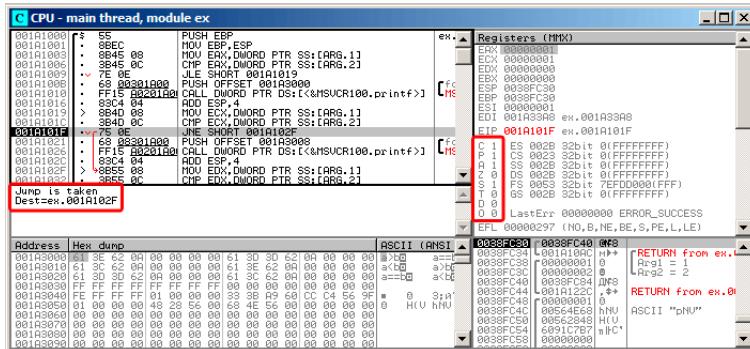


Figura 12.5: OllyDbg: f_signed():

JGE SF=OF, :

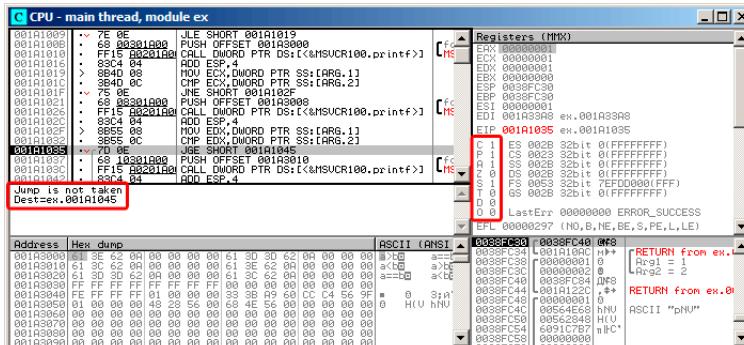


Figura 12.6: OllyDbg: f_signed():

x86 + MSVC + Hiew

f_unsigned() «a==b», . Hiew:

The screenshot shows the Hiew debugger interface with the assembly dump window open. The assembly code for the `f_unsigned()` function is displayed. The code compares the values of `a` and `b` using `cmp` instructions and prints the result using `printf`. The assembly listing includes comments indicating the comparison and the output string. The memory dump and registers panes are also visible at the bottom of the window.

```

C:\Polygon\ollydbg\7_1.exe      @FRO ----- a32 PE .00401000|Hiew 8.02 (c)SEN
.00401000: 55                 push    ebp
.00401001: 88EC               mov     ebp,esp
.00401003: 8B4508             mov     eax,[ebp][8]
.00401006: 3B450C             cmp     eax,[ebp][00C]
.00401009: 7E0D               jle    .000401018 --@1
.0040100B: E800B04000         push    00040B000 --@2
.00401010: E8AA000000         call    .0004010BF --@3
.00401015: 83C404             add    esp,4
.00401018: 8B4D08             imov   ecx,[ebp][8]
.0040101B: 3B4D0C             cmp    ecx,[ebp][00C]
.0040101E: 750D               jnz    .00040102D --@4
.00401020: E800B04000         push    00040B008 ;'a==b' --@5
.00401025: E895000000         call    .0004010BF --@3
.0040102A: 83C404             add    esp,4
.0040102D: 8B5508             4mov   edx,[ebp][8]
.00401030: 3B550C             cmp    edx,[ebp][00C]
.00401033: 700D               jge    .000401042 --@6
.00401035: E810B04000         push    00040B010 --@7
.0040103A: E880000000         call    .0004010BF --@3
.0040103F: 83C404             add    esp,4
.00401042: 5D                 6pop   ebp
.00401043: C3                 retn  ; _^_~_~_~_~_~_~_~_~_~_~_~_~_~_~_~_
.00401044: CC                 int    3
.00401045: CC                 int    3
.00401046: CC                 int    3
.00401047: CC                 int    3
.00401048: CC                 int    3

```

Figura 12.7: Hiew: f_unsigned()

printf(), «a==b».

- ;
- ;
- .
- JMP,
-
-

:

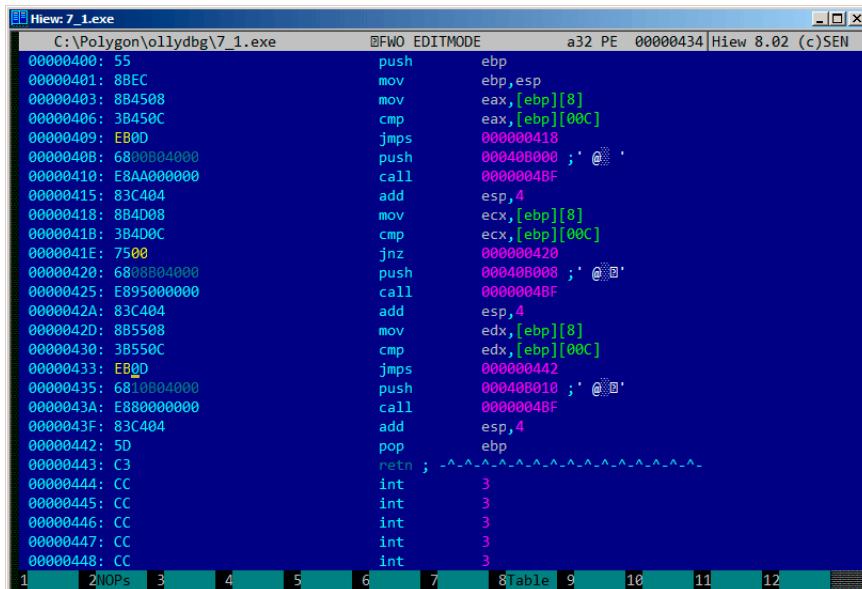


Figura 12.8: Hiew: f_unsigned()

Sin optimización GCC

Sin optimización GCC 4.4.1 puts() ([3.4.3 on page 21](#)) printf().

Con optimización GCC

?

Listing 12.4: GCC 4.8.1 f_signed()

```
f_signed:
    mov     eax, DWORD PTR [esp+8]
    cmp     DWORD PTR [esp+4], eax
    jg      .L6
    je      .L7
    jge     .L1
    mov     DWORD PTR [esp+4], OFFSET FLAT:.LC2 ; "a<b"
    jmp     puts
.L6:
    mov     DWORD PTR [esp+4], OFFSET FLAT:.LC0 ; "a>b"
    jmp     puts
.L1:
```

```

rep ret
.L7:
    mov     DWORD PTR [esp+4], OFFSET FLAT:.LC1 ; "a==b"
    jmp     puts

```

JMP puts CALL puts / RETN.:[13.1.1 on page 184](#).

. MSVC 2012. Jcc

f_unsigned():

Listing 12.5: GCC 4.8.1 f_unsigned()

```

f_unsigned:
    push    esi
    push    ebx
    sub     esp, 20
    mov     esi, DWORD PTR [esp+32]
    mov     ebx, DWORD PTR [esp+36]
    cmp     esi, ebx
    ja      .L13
    cmp     esi, ebx          ;
    je      .L14
.L10:
    jb      .L15
    add    esp, 20
    pop    ebx
    pop    esi
    ret
.L15:
    mov     DWORD PTR [esp+32], OFFSET FLAT:.LC2 ; "a<b"
    add    esp, 20
    pop    ebx
    pop    esi
    jmp     puts
.L13:
    mov     DWORD PTR [esp], OFFSET FLAT:.LC0 ; "a>b"
    call    puts
    cmp     esi, ebx
    jne     .L10
.L14:
    mov     DWORD PTR [esp+32], OFFSET FLAT:.LC1 ; "a==b"
    add    esp, 20
    pop    ebx
    pop    esi
    jmp     puts

```

12.1.2 ARM

32- ARM

Con optimización Keil 6/2013 (Modo ARM)

Listing 12.6: Con optimización Keil 6/2013 (Modo ARM)

```
.text:000000B8          EXPORT f_signed
.text:000000B8          f_signed           ; CODE XREF: ↴
    ↴ main+C
.text:000000B8 70 40 2D E9      STMFD   SP!, {R4-R6,LR}
.text:000000BC 01 40 A0 E1      MOV     R4, R1
.text:000000C0 04 00 50 E1      CMP     R0, R4
.text:000000C4 00 50 A0 E1      MOV     R5, R0
.text:000000C8 1A 0E 8F C2      ADRGT   R0, aAB           ; "a>
    ↴ >b\n"
.text:000000CC A1 18 00 CB      BLGT    __2printf
.text:000000D0 04 00 55 E1      CMP     R5, R4
.text:000000D4 67 0F 8F 02      ADREQ   R0, aAB_0         ; "a<
    ↴ ==b\n"
.text:000000D8 9E 18 00 0B      BLEQ    __2printf
.text:000000DC 04 00 55 E1      CMP     R5, R4
.text:000000E0 70 80 BD A8      LDMGEFD SP!, {R4-R6,PC}
.text:000000E4 70 40 BD E8      LDMFD   SP!, {R4-R6,LR}
.text:000000E8 19 0E 8F E2      ADR     R0, aAB_1         ; "a<
    ↴ <b\n"
.text:000000EC 99 18 00 EA      B       __2printf
.text:000000EC          ; End of function f_signed
```

(condition field).

(Greater Than). ADRGT a>b\n printf().

(Greater or Equal).

LDMGEFD SP!, {R4-R6,PC}

printf() «a<b\n». «printf() con varios argumentos» (6.2.1 on page 60).

f_unsigned ADRHI, BLHI, y LDMCSFD (HI = Unsigned higher, CS = Carry Set (greater than or equal))

Listing 12.7: main()

```
.text:00000128          EXPORT main
.text:00000128          main
.text:00000128 10 40 2D E9      STMFD   SP!, {R4,LR}
.text:0000012C 02 10 A0 E3      MOV     R1, #2
.text:00000130 01 00 A0 E3      MOV     R0, #1
```

```
.text:00000134 DF FF FF EB      BL      f_signed
.text:00000138 02 10 A0 E3      MOV     R1, #2
.text:0000013C 01 00 A0 E3      MOV     R0, #1
.text:00000140 EA FF FF EB      BL      f_unsigned
.text:00000144 00 00 A0 E3      MOV     R0, #0
.text:00000148 10 80 BD E8      LDMFD   SP!, {R4,PC}
.text:00000148                 ; End of function main
```

[: 33.1 on page 589.](#)

Con optimización Keil 6/2013 (Modo Thumb)

Listing 12.8: Con optimización Keil 6/2013 (Modo Thumb)

```
.text:00000072          f_signed ; CODE XREF: main+6
.text:00000072 70 B5      PUSH    {R4-R6,LR}
.text:00000074 0C 00      MOVS    R4, R1
.text:00000076 05 00      MOVS    R5, R0
.text:00000078 A0 42      CMP     R0, R4
.text:0000007A 02 DD      BLE    loc_82
.text:0000007C A4 A0      ADR     R0, aAB           ; "a>b\n"
.text:0000007E 06 F0 B7 F8  BL      __2printf
.text:00000082
.text:00000082          loc_82 ; CODE XREF: f_signed+8
.text:00000082 A5 42      CMP     R5, R4
.text:00000084 02 D1      BNE    loc_8C
.text:00000086 A4 A0      ADR     R0, aAB_0         ; "a==b\n"
.text:00000088 06 F0 B2 F8  BL      __2printf
.text:0000008C
.text:0000008C          loc_8C ; CODE XREF: f_signed+12
.text:0000008C A5 42      CMP     R5, R4
.text:0000008E 02 DA      BGE    locret_96
.text:00000090 A3 A0      ADR     R0, aAB_1         ; "a<b\n"
.text:00000092 06 F0 AD F8  BL      __2printf
.text:00000096
.text:00000096          locret_96 ; CODE XREF: f_signed+1C
.text:00000096 70 BD      POP    {R4-R6,PC}
.text:00000096          ; End of function f_signed
```

BLE *Less than or Equal*, BNENot Equal, BGEGreater than or Equal.

f_unsigned BLS (*Unsigned lower or same*) y BCS (*Carry Set (Greater than or equal)*).

ARM64: Con optimización GCC (Linaro) 4.9

Listing 12.9: f_signed()

```

f_signed:
; W0=a, W1=b
    cmp    w0, w1
    bgt   .L19    ; Branch if Greater Than (a>b)
    beq   .L20    ; Branch if Equal (a==b)
    bge   .L15    ; Branch if Greater than or Equal (a>=b)
    ↵ ) ()
    ; a<b
    adrp   x0, .LC11      ; "a<b"
    add    x0, x0, :lo12:.LC11
    b      puts
.L19:
    adrp   x0, .LC9       ; "a>b"
    add    x0, x0, :lo12:.LC9
    b      puts
.L15:
    ;
    ret
.L20:
    adrp   x0, .LC10      ; "a==b"
    add    x0, x0, :lo12:.LC10
    b      puts

```

Listing 12.10: f_unsigned()

```

f_unsigned:
    stp    x29, x30, [sp, -48]!
; W0=a, W1=b
    cmp    w0, w1
    add    x29, sp, 0
    str    x19, [sp,16]
    mov    w19, w0
    bhi   .L25    ; Branch if HIgher (a>b)
    cmp    w19, w1
    beq   .L26    ; Branch if Equal (a==b)
.L23:
    bcc   .L27    ; Branch if Carry Clear () (a<b)
;
    ldr    x19, [sp,16]
    ldp    x29, x30, [sp], 48
    ret
.L27:
    ldr    x19, [sp,16]
    adrp   x0, .LC11      ; "a<b"
    ldp    x29, x30, [sp], 48

```

```

        add    x0, x0, :lo12:.LC11
        b      puts
.L25:
        adrp   x0, .LC9          ; "a>b"
        str    x1, [x29,40]
        add    x0, x0, :lo12:.LC9
        bl     puts
        ldr    x1, [x29,40]
        cmp    w19, w1
        bne   .L23   ; Branch if Not Equal
.L26:
        ldr    x19, [sp,16]
        adrp   x0, .LC10         ; "a==b"
        ldp    x29, x30, [sp], 48
        add    x0, x0, :lo12:.LC10
        b      puts

```

Ejercicio

12.1.3 MIPS

Listing 12.11: Sin optimización GCC 4.4.5 (IDA)

```

.text:00000000 f_signed:                      # CODE ↴
    ↳ XREF: main+18
.text:00000000
.text:00000000 var_10             = -0x10
.text:00000000 var_8              = -8
.text:00000000 var_4              = -4
.text:00000000 arg_0              = 0
.text:00000000 arg_4              = 4
.text:00000000
.text:00000000 addiu   $sp, -0x20
.text:00000004 sw      $ra, 0x20+var_4($sp)
.text:00000008 sw      $fp, 0x20+var_8($sp)
.text:0000000C move   $fp, $sp
.text:00000010 la      $gp, __gnu_local_gp
.text:00000018 sw      $gp, 0x20+var_10($sp)
; :
.text:0000001C sw      $a0, 0x20+arg_0($fp)
.text:00000020 sw      $a1, 0x20+arg_4($fp)
; reload them.

```

.text:00000024	lw	\$v1, 0x20+arg_0(\$fp)
.text:00000028	lw	\$v0, 0x20+arg_4(\$fp)
; \$v0=b		
; \$v1=a		
.text:0000002C	or	\$at, \$zero ; NOP
; "slt \$v0,\$v0,\$v1" .		
; \$v0 \$v0<\$v1 (b<a) :		
.text:00000030	slt	\$v0, \$v1
; .		
; "beq \$v0,\$zero,loc_5c" :		
.text:00000034	beqz	\$v0, loc_5C
;		
.text:00000038	or	\$at, \$zero ; branch ↗
↳ delay slot, NOP		
.text:0000003C	lui	\$v0, (unk_230 >> 16) # "↗
↳ a>b"		
.text:00000040	addiu	\$a0, \$v0, (unk_230 & 0x
↳ xFFFF) # "a>b"		
.text:00000044	lw	\$v0, (puts & 0xFFFF)(\$gp↗
↳)		
.text:00000048	or	\$at, \$zero ; NOP
.text:0000004C	move	\$t9, \$v0
.text:00000050	jalr	\$t9
.text:00000054	or	\$at, \$zero ; branch ↗
↳ delay slot, NOP		
.text:00000058	lw	\$gp, 0x20+var_10(\$fp)
.text:0000005C		# CODE ↗
↳ XREF: f_signed+34		
.text:0000005C	lw	\$v1, 0x20+arg_0(\$fp)
.text:00000060	lw	\$v0, 0x20+arg_4(\$fp)
.text:00000064	or	\$at, \$zero ; NOP
; :		
.text:00000068	bne	\$v1, \$v0, loc_90
.text:0000006C	or	\$at, \$zero ; branch ↗
↳ delay slot, NOP		
; :		
.text:00000070	lui	\$v0, (aAB >> 16) # "a==↗
↳ b"		
.text:00000074	addiu	\$a0, \$v0, (aAB & 0xFFFF)↗
↳ # "a==b"		
.text:00000078	lw	\$v0, (puts & 0xFFFF)(\$gp↗
↳)		
.text:0000007C	or	\$at, \$zero ; NOP
.text:00000080	move	\$t9, \$v0
.text:00000084	jalr	\$t9

```

.text:00000088          or      $at, $zero ; branch ↵
    ↳ delay slot, NOP
.text:0000008C          lw      $gp, 0x20+var_10($fp)
.text:00000090
.text:00000090 loc_90:           # CODE ↵
    ↳ XREF: f_signed+68
.text:00000090          lw      $v1, 0x20+arg_0($fp)
.text:00000094          lw      $v0, 0x20+arg_4($fp)
.text:00000098          or      $at, $zero ; NOP
; $v1<$v0 (a<b), $v0 :
.text:0000009C          slt     $v0, $v1, $v0
; :
.text:000000A0          beqz   $v0, loc_C8
.text:000000A4          or      $at, $zero ; branch ↵
    ↳ delay slot, NOP
;
.text:000000A8          lui     $v0, (aAB_0 >> 16) # "a"
    ↳ <b"
.text:000000AC          addiu  $a0, $v0, (aAB_0 & 0
    ↳xFFFF) # "a<b"
.text:000000B0          lw      $v0, (puts & 0xFFFF)($gp)
    ↳ )
.text:000000B4          or      $at, $zero ; NOP
.text:000000B8          move   $t9, $v0
.text:000000BC          jalr   $t9
.text:000000C0          or      $at, $zero ; branch ↵
    ↳ delay slot, NOP
.text:000000C4          lw      $gp, 0x20+var_10($fp)
.text:000000C8
; :
.text:000000C8 loc_C8:           # CODE ↵
    ↳ XREF: f_signed+A0
.text:000000C8          move   $sp, $fp
.text:000000CC          lw      $ra, 0x20+var_4($sp)
.text:000000D0          lw      $fp, 0x20+var_8($sp)
.text:000000D4          addiu $sp, 0x20
.text:000000D8          jr      $ra
.text:000000DC          or      $at, $zero ; branch ↵
    ↳ delay slot, NOP
.text:000000DC # End of function f_signed

```

«SLT REG0, REG0, REG1» «SLT REG0, REG1».

Listing 12.12: Sin optimización GCC 4.4.5 (IDA)

```

.text:000000E0 f_unsigned:           # CODE ↵
    ↳ XREF: main+28
.text:000000E0

```

```

.text:000000E0 var_10          = -0x10
.text:000000E0 var_8           = -8
.text:000000E0 var_4           = -4
.text:000000E0 arg_0           = 0
.text:000000E0 arg_4           = 4
.text:000000E0
.text:000000E0                 addiu   $sp, -0x20
.text:000000E4                 sw      $ra, 0x20+var_4($sp)
.text:000000E8                 sw      $fp, 0x20+var_8($sp)
.text:000000EC                 move    $fp, $sp
.text:000000F0                 la      $gp, __gnu_local_gp
.text:000000F8                 sw      $gp, 0x20+var_10($sp)
.text:000000FC                 sw      $a0, 0x20+arg_0($fp)
.text:00000100                 sw      $a1, 0x20+arg_4($fp)
.text:00000104                 lw      $v1, 0x20+arg_0($fp)
.text:00000108                 lw      $v0, 0x20+arg_4($fp)
.text:0000010C                 or      $at, $zero
.text:00000110                 sltu   $v0, $v1
.text:00000114                 beqz  $v0, loc_13C
.text:00000118                 or      $at, $zero
.text:0000011C                 lui    $v0, (unk_230 >> 16)
.text:00000120                 addiu $a0, $v0, (unk_230 & 0x
    ↴ xFFFF)
.text:00000124                 lw      $v0, (puts & 0xFFFF)($gp)
    ↴ )
.text:00000128                 or      $at, $zero
.text:0000012C                 move   $t9, $v0
.text:00000130                 jalr   $t9
.text:00000134                 or      $at, $zero
.text:00000138                 lw      $gp, 0x20+var_10($fp)
.text:0000013C loc_13C:          # CODE ↴
    ↴ XREF: f_unsigned+34
.text:0000013C                 lw      $v1, 0x20+arg_0($fp)
.text:00000140                 lw      $v0, 0x20+arg_4($fp)
.text:00000144                 or      $at, $zero
.text:00000148                 bne   $v1, $v0, loc_170
.text:0000014C                 or      $at, $zero
.text:00000150                 lui    $v0, (aAB >> 16) # "a=="
    ↴ b"
.text:00000154                 addiu $a0, $v0, (aAB & 0xFFFF)
    ↴ # "a==b"
.text:00000158                 lw      $v0, (puts & 0xFFFF)($gp)
    ↴ )
.text:0000015C                 or      $at, $zero
.text:00000160                 move   $t9, $v0
.text:00000164                 jalr   $t9

```

```

.text:00000168          or      $at, $zero
.text:0000016C          lw      $gp, 0x20+var_10($fp)
.text:00000170
.text:00000170 loc_170:           # CODE ↴
    ↳ XREF: f_unsigned+68
.text:00000170          lw      $v1, 0x20+arg_0($fp)
.text:00000174          lw      $v0, 0x20+arg_4($fp)
.text:00000178          or      $at, $zero
.text:0000017C          sltu   $v0, $v1, $v0
.text:00000180          beqz   $v0, loc_1A8
.text:00000184          or      $at, $zero
.text:00000188          lui     $v0, (aAB_0 >> 16) # "a>
    ↳ <b"
.text:0000018C          addiu  $a0, $v0, (aAB_0 & 0x
    ↳ xFFFF) # "a<b"
.text:00000190          lw      $v0, (puts & 0xFFFF)($gp)
    ↳ )
.text:00000194          or      $at, $zero
.text:00000198          move   $t9, $v0
.text:0000019C          jalr   $t9
.text:000001A0          or      $at, $zero
.text:000001A4          lw      $gp, 0x20+var_10($fp)
.text:000001A8
.text:000001A8 loc_1A8:           # CODE ↴
    ↳ XREF: f_unsigned+A0
.text:000001A8          move   $sp, $fp
.text:000001AC          lw      $ra, 0x20+var_4($sp)
.text:000001B0          lw      $fp, 0x20+var_8($sp)
.text:000001B4          addiu $sp, 0x20
.text:000001B8          jr     $ra
.text:000001BC          or      $at, $zero
.text:000001BC # End of function f_unsigned

```

12.2

```

int my_abs (int i)
{
    if (i<0)
        return -i;
    else
        return i;
}

```

12.2.1 Con optimización MSVC

Listing 12.13: Con optimización MSVC 2012 x64

```
i$ = 8
my_abs PROC
; ECX = input
    test    ecx,  ecx
;
;
    jns     SHORT $LN2@my_abs
;
    neg    ecx
$LN2@my_abs:
; EAX:
    mov    eax,  ecx
    ret    0
my_abs ENDP
```

12.2.2 Con optimización Keil 6/2013: Modo Thumb

Listing 12.14: Con optimización Keil 6/2013: Modo Thumb

```
my_abs PROC
    CMP    r0,#0
; ?
;
    BGE    |L0.6|
; 0:
    RSBS   r0,r0,#0
|L0.6|
    BX     lr
ENDP
```

12.2.3 Con optimización Keil 6/2013: Modo ARM

Listing 12.15: Con optimización Keil 6/2013: Modo ARM

```
my_abs PROC
    CMP    r0,#0
; "Reverse Subtract" :
    RSBLT  r0,r0,#0
    BX     lr
ENDP
```

[33.1 on page 589.](#)

12.2.4 Sin optimización GCC 4.9 (ARM64)

Listing 12.16: Con optimización GCC 4.9 (ARM64)

```
my_abs:
    sub    sp, sp, #16
    str    w0, [sp,12]
    ldr    w0, [sp,12]
;
;
    cmp    w0, wzr
    bge   .L2
    ldr    w0, [sp,12]
    neg    w0, w0
    b     .L3
.L2:
    ldr    w0, [sp,12]
.L3:
    add    sp, sp, 16
    ret
```

12.2.5 MIPS

Listing 12.17: Con optimización GCC 4.4.5 (IDA)

```
my_abs:
; $a0<0:
; ($a0) $v0:
        bltz    $a0, locret_10
        move    $v0, $a0
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP
locret_10:
; $v0:
        jr     $ra
; "subu $v0,$zero,$a0" ($v0=0-$a0)
        negu    $v0, $a0
```

: BLTZ («Branch if Less Than Zero»).

12.2.6 ?

[45 on page 663.](#)

C/C++:

```
expression ? expression : expression
```

:

```
const char* f (int a)
{
    return a==10 ? "it is ten" : "it is not ten";
};
```

12.3.1 x86

Listing 12.18: Sin optimización MSVC 2008

```
$SG746 DB      'it is ten', 00H
$SG747 DB      'it is not ten', 00H

tv65 = -4 ;
_a$ = 8
_f PROC
    push    ebp
    mov     ebp, esp
    push    ecx
; 10
    cmp     DWORD PTR _a$[ebp], 10
; $LN3@f
    jne     SHORT $LN3@f
; :
    mov     DWORD PTR tv65[ebp], OFFSET $SG746 ; 'it is ten'
    jmp     SHORT $LN4@f
$LN3@f:
; :
    mov     DWORD PTR tv65[ebp], OFFSET $SG747 ; 'it is not'
    ten'
$LN4@f:
;
    mov     eax, DWORD PTR tv65[ebp]
    mov     esp, ebp
    pop     ebp
    ret     0
_f    ENDP
```

Listing 12.19: Con optimización MSVC 2008

```
$SG792 DB      'it is ten', 00H
$SG793 DB      'it is not ten', 00H

_a$ = 8 ; size = 4
_f      PROC
; 10
    cmp     DWORD PTR _a$[esp-4], 10
    mov     eax, OFFSET $SG792 ; 'it is ten'
; $LN4@f
    je      SHORT $LN4@f
    mov     eax, OFFSET $SG793 ; 'it is not ten'
$LN4@f:
    ret     0
_f      ENDP
```

:

Listing 12.20: Con optimización MSVC 2012 x64

```
$SG1355 DB      'it is ten', 00H
$SG1356 DB      'it is not ten', 00H

a$ = 8
_f      PROC
;
    lea     rdx, OFFSET FLAT:$SG1355 ; 'it is ten'
    lea     rax, OFFSET FLAT:$SG1356 ; 'it is not ten'
; 10
    cmp     ecx, 10
; ("it is ten")
;
    cmove   rax, rdx
    ret     0
_f      ENDP
```

Con optimización GCC 4.8 para x86 CMOVcc.

12.3.2 ARM

Con optimización Keil para:

Listing 12.21: Con optimización Keil 6/2013 (Modo ARM)

```
f PROC
; 10
    CMP      r0,#0xa
;
    ADREQ   r0,|L0.16| ; "it is ten"
;
    ADRNE   r0,|L0.28| ; "it is not ten"
    BX      lr
    ENDP

|L0.16|
    DCB      "it is ten",0
|L0.28|
    DCB      "it is not ten",0
```

Con optimización Keil para:

Listing 12.22: Con optimización Keil 6/2013 (Modo Thumb)

```
f PROC
; 10
    CMP      r0,#0xa
; |L0.8| Equal
    BEQ      |L0.8|
    ADR      r0,|L0.12| ; "it is not ten"
    BX      lr
|L0.8|
    ADR      r0,|L0.28| ; "it is ten"
    BX      lr
    ENDP

|L0.12|
    DCB      "it is not ten",0
|L0.28|
    DCB      "it is ten",0
```

12.3.3 ARM64

Con optimización GCC (Linaro) 4.9 para ARM64 :

Listing 12.23: Con optimización GCC (Linaro) 4.9

```
f:
    cmp      x0, 10
    beq      .L3           ; branch if equal
```

```

        adrp    x0, .LC1          ; "it is ten"
        add     x0, x0, :lo12:.LC1
        ret

.L3:
        adrp    x0, .LC0          ; "it is not ten"
        add     x0, x0, :lo12:.LC0
        ret

.LC0:
        .string "it is ten"
.LC1:
        .string "it is not ten"

```

[ARM13a, p390, C5.5].

12.3.4 MIPS

:

Listing 12.24: Con optimización GCC 4.4.5 (salida de assembly)

```

$LC0:
        .ascii  "it is not ten\000"
$LC1:
        .ascii  "it is ten\000"
f:
        li      $2,10                  # 0xa
; $a0 10, :
        beq    $4,$2,$L2
        nop ; branch delay slot

; :
        lui    $2,%hi($LC0)
        j     $31
        addiu $2,$2,%lo($LC0)

$L2:
; :
        lui    $2,%hi($LC1)
        j     $31
        addiu $2,$2,%lo($LC1)

```

12.3.5

```

const char* f (int a)
{

```

```

    if (a==10)
        return "it is ten";
    else
        return "it is not ten";
};

```

Listing 12.25: Con optimización GCC 4.8

```

.LC0:
    .string "it is ten"
.LC1:
    .string "it is not ten"
f:
.LFB0:
; 10
    cmp     DWORD PTR [esp+4], 10
    mov     edx, OFFSET FLAT:.LC1 ; "it is not ten"
    mov     eax, OFFSET FLAT:.LC0 ; "it is ten"
;
;
    cmovne eax, edx
    ret

```

Con optimización Keil en Spanish text placeholder.[12.21](#).

MSVC 2012 .

12.3.6 Conclusión

[33.1 on page 589](#).

12.4

12.4.1 32-bit

```

int my_max(int a, int b)
{
    if (a>b)
        return a;
    else
        return b;
};

int my_min(int a, int b)
{

```

```

    if (a<b)
        return a;
    else
        return b;
};

```

Listing 12.26: Sin optimización MSVC 2013

```

_a$ = 8
_b$ = 12
_my_min PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
; :
    cmp     eax, DWORD PTR _b$[ebp]
; :
    jge     SHORT $LN2@my_min
;
    mov     eax, DWORD PTR _a$[ebp]
    jmp     SHORT $LN3@my_min
    jmp     SHORT $LN3@my_min ; JMP
$LN2@my_min:
; B
    mov     eax, DWORD PTR _b$[ebp]
$LN3@my_min:
    pop    ebp
    ret    0
_my_min ENDP

_a$ = 8
_b$ = 12
_my_max PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
; :
    cmp     eax, DWORD PTR _b$[ebp]
; :
    jle     SHORT $LN2@my_max
;
    mov     eax, DWORD PTR _a$[ebp]
    jmp     SHORT $LN3@my_max
    jmp     SHORT $LN3@my_max ; JMP
$LN2@my_max:
; B
    mov     eax, DWORD PTR _b$[ebp]

```

```
$LN3@my_max:
    pop      ebp
    ret      0
_my_max ENDP
```

Listing 12.27: Con optimización Keil 6/2013 (Modo Thumb)

```
my_max PROC
; R0=A
; R1=B
; :
    CMP      r0,r1
; :
    BGT      |L0.6|
; (A<=B) R1 (B):
    MOVS     r0,r1
|L0.6|
;
    BX       lr
    ENDP

my_min PROC
; R0=A
; R1=B
; :
    CMP      r0,r1
; :
    BLT      |L0.14|
; (A>=B) R1 (B):
    MOVS     r0,r1
|L0.14|
;
    BX       lr
    ENDP
```

Listing 12.28: Con optimización Keil 6/2013 (Modo ARM)

```
my_max PROC
; R0=A
; R1=B
; :
    CMP      r0,r1
;
```

```

; ; ; ; ;
        MOVLE    r0,r1
        BX      lr
        ENDP

my_min PROC
; R0=A
; R1=B
; :
        CMP      r0,r1
;
;
;
        MOVGE    r0,r1
        BX      lr
        ENDP

```

Con optimización GCC 4.8.1 MSVC 2013

Listing 12.29: Con optimización MSVC 2013

```

my_max:
        mov      edx, DWORD PTR [esp+4]
        mov      eax, DWORD PTR [esp+8]
; EDX=A
; EAX=B
; :
        cmp      edx, eax
; A>=B,
; ( A<B)
        cmovge  eax, edx
        ret

my_min:
        mov      edx, DWORD PTR [esp+4]
        mov      eax, DWORD PTR [esp+8]
; EDX=A
; EAX=B
; :
        cmp      edx, eax
; A<=B,
; ( A>B)
        cmovle  eax, edx
        ret

```

12.4.2 64-bit

```
#include <stdint.h>

int64_t my_max(int64_t a, int64_t b)
{
    if (a>b)
        return a;
    else
        return b;
};

int64_t my_min(int64_t a, int64_t b)
{
    if (a<b)
        return a;
    else
        return b;
};
```

Listing 12.30: Sin optimización GCC 4.9.1 ARM64

```
my_max:
    sub    sp, sp, #16
    str    x0, [sp,8]
    str    x1, [sp]
    ldr    x1, [sp,8]
    ldr    x0, [sp]
    cmp    x1, x0
    ble   .L2
    ldr    x0, [sp,8]
    b     .L3
.L2:
    ldr    x0, [sp]
.L3:
    add    sp, sp, 16
    ret

my_min:
    sub    sp, sp, #16
    str    x0, [sp,8]
    str    x1, [sp]
    ldr    x1, [sp,8]
    ldr    x0, [sp]
    cmp    x1, x0
    bge   .L5
    ldr    x0, [sp,8]
    b     .L6
```

```
.L5:
    ldr      x0, [sp]
.L6:
    add      sp, sp, 16
    ret
```

Listing 12.31: Con optimización GCC 4.9.1 x64

```
my_max:
; RDI=A
; RSI=B
; :
    cmp      rdi, rsi
; :
    mov      rax, rsi
; A>=B, A (RDI)  RAX .
; ( A<B)
    cmovge  rax, rdi
    ret

my_min:
; RDI=A
; RSI=B
; :
    cmp      rdi, rsi
; :
    mov      rax, rsi
; A<=B, A (RDI)  RAX .
; ( A>B)
    cmovle  rax, rdi
    ret
```

MSVC 2013 .

«Conditional SELect».

Listing 12.32: Con optimización GCC 4.9.1 ARM64

```
my_max:
; X0=A
; X1=B
; :
    cmp      x0, x1
;
```

```

;    A<B
        csel    x0, x0, x1, ge
        ret

my_min:
; X0=A
; X1=B
;
;      cmp     x0, x1
;
;    A>B
        csel    x0, x0, x1, le
        ret

```

12.4.3 MIPS

:

Listing 12.33: Con optimización GCC 4.4.5 (IDA)

```

my_max:
; $v1 $a1<$a0:
        slt    $v1, $a1, $a0
; $a1<$a0:
        beqz   $v1, locret_10
; branch delay slot
; $a1 $v0 :
        move   $v0, $a1
; $a0 $v0:
        move   $v0, $a0

locret_10:
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP

;
my_min:
        slt    $v1, $a0, $a1
        beqz   $v1, locret_28
        move   $v0, $a1
        move   $v0, $a0

locret_28:
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP

```

12.5 Conclusión

12.5.1 x86

:

Listing 12.34: x86

```
CMP register, register/value
Jcc true ; cc=
false:
...
...
JMP exit
true:
...
...
exit:
```

12.5.2 ARM

Listing 12.35: ARM

```
CMP register, register/value
Bcc true ; cc=
false:
...
...
JMP exit
true:
...
...
exit:
```

12.5.3 MIPS

Listing 12.36:

```
BEQZ REG, label
...
```

Listing 12.37:

```
BLTZ REG, label
...
```

Listing 12.38:

```
BEQ REG1, REG2, label
...

```

Listing 12.39:

```
BNE REG1, REG2, label
...

```

Listing 12.40:

```
SLT REG1, REG2, REG3
BEQ REG1, label
...

```

Listing 12.41:

```
SLTU REG1, REG2, REG3
BEQ REG1, label
...

```

12.5.4**ARM****Listing 12.42: ARM (Modo ARM)**

```
CMP register, register/value
instr1_cc ;
instr2_cc ;
...
...
```

[: 17.7.2 on page 318.](#)

Listing 12.43: ARM (Modo Thumb)

```
CMP register, register/value
ITEEE EQ ; : if-then-else-else-else
instr1    ;
instr2    ;
instr3    ;
instr4    ;
```

12.6 Ejercicio

(ARM64) Spanish text placeholder.[12.23](#).

Capítulo 13

switch() / case / default

13.1

```
#include <stdio.h>

void f (int a)
{
    switch (a)
    {
        case 0: printf ("zero\n"); break;
        case 1: printf ("one\n"); break;
        case 2: printf ("two\n"); break;
        default: printf ("something unknown\n"); break;
    };
}

int main()
{
    f (2); // test
}
```

13.1.1 x86

Sin optimización MSVC

(MSVC 2010):

Listing 13.1: MSVC 2010

```
tv64 = -4 ; size = 4
```

```

_a$ = 8    ; size = 4
_f      PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]
    mov     DWORD PTR tv64[ebp], eax
    cmp     DWORD PTR tv64[ebp], 0
    je      SHORT $LN4@f
    cmp     DWORD PTR tv64[ebp], 1
    je      SHORT $LN3@f
    cmp     DWORD PTR tv64[ebp], 2
    je      SHORT $LN2@f
    jmp     SHORT $LN1@f
$LN4@f:
    push    OFFSET $SG739 ; 'zero', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN7@f
$LN3@f:
    push    OFFSET $SG741 ; 'one', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN7@f
$LN2@f:
    push    OFFSET $SG743 ; 'two', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN7@f
$LN1@f:
    push    OFFSET $SG745 ; 'something unknown', 0aH, 00H
    call    _printf
    add    esp, 4
$LN7@f:
    mov     esp, ebp
    pop    ebp
    ret    0
_f      ENDP

```

```

void f (int a)
{
    if (a==0)
        printf ("zero\n");
    else if (a==1)
        printf ("one\n");
    else if (a==2)
        printf ("two\n");

```

```

    else
        printf ("something unknown\n");
};

```

Con optimización MSVC

MSVC (/Ox): cl 1.c /Fa1.asm /Ox

Listing 13.2: MSVC

```

_a$ = 8 ; size = 4
_f    PROC
    mov    eax, DWORD PTR _a$[esp-4]
    sub    eax, 0
    je     SHORT $LN4@f
    sub    eax, 1
    je     SHORT $LN3@f
    sub    eax, 1
    je     SHORT $LN2@f
    mov    DWORD PTR _a$[esp-4], OFFSET $SG791 ; 'something '
    ↴ unknown', 0aH, 00H
    jmp    _printf
$LN2@f:
    mov    DWORD PTR _a$[esp-4], OFFSET $SG789 ; 'two', 0aH, 00
    ↴ H
    jmp    _printf
$LN3@f:
    mov    DWORD PTR _a$[esp-4], OFFSET $SG787 ; 'one', 0aH, 00
    ↴ H
    jmp    _printf
$LN4@f:
    mov    DWORD PTR _a$[esp-4], OFFSET $SG785 ; 'zero', 0aH, 00
    ↴ 00H
    jmp    _printf
_f    ENDP

```

- ESP RA
- ESP+4

«printf() con varios argumentos» ([6.2.1 on page 60](#)).

OllyDbg

OllyDbg.

OllyDbg. EAX 2, :

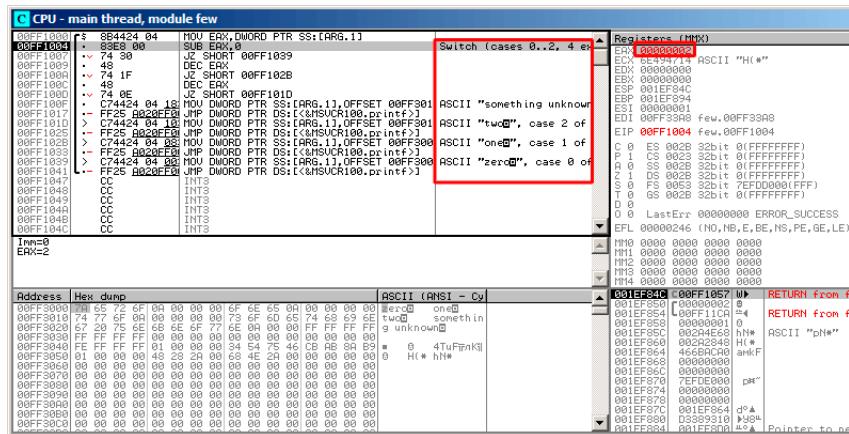


Figura 13.1: OllyDbg: EAX

0 2 en EAX., EAX 2. ZF 0.:

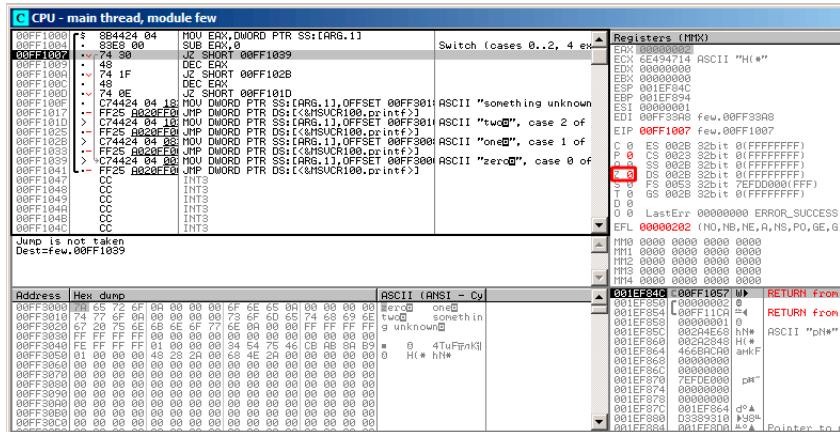


Figura 13.2: OllyDbg: SUB

DEC EAX 1. 1 ZF 0:

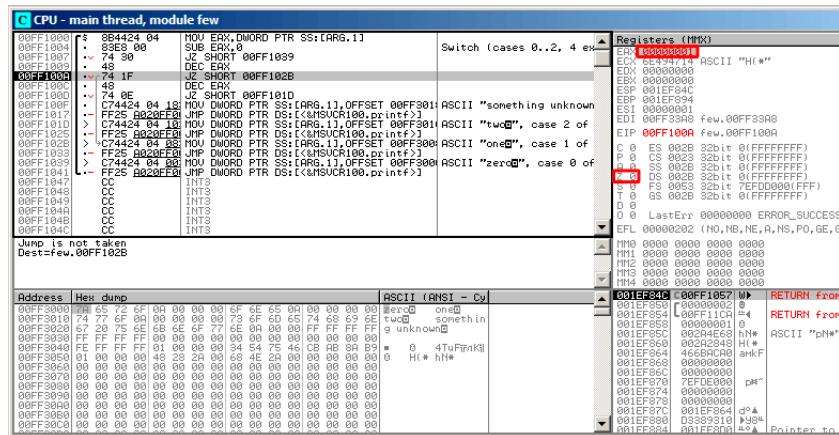


Figura 13.3: OllyDbg: DEC

DEC. EAX 0 ZF:

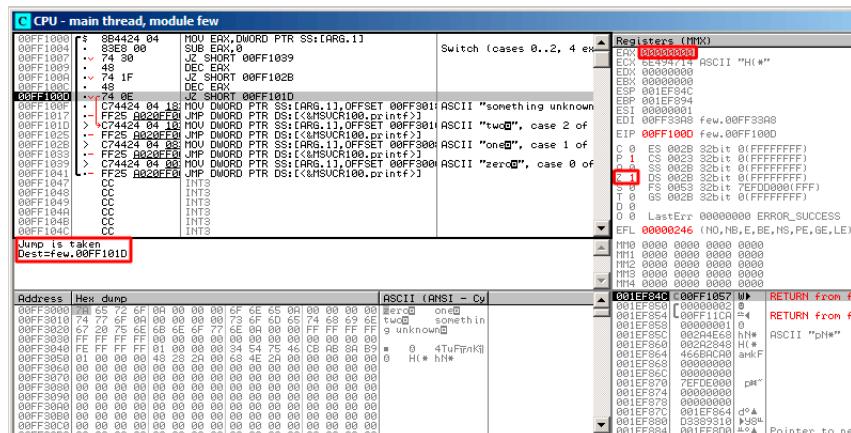


Figura 13.4: OllyDbg: DEC

OllyDbg

«[TWO]» :

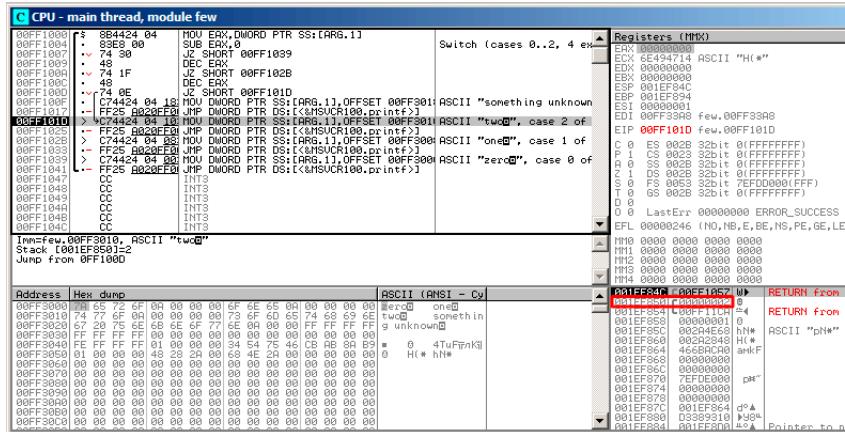


Figura 13.5: OllyDbg:

0x001EF850.

MOV 0x001EF850 . . printf() MSVCR100.DLL ()

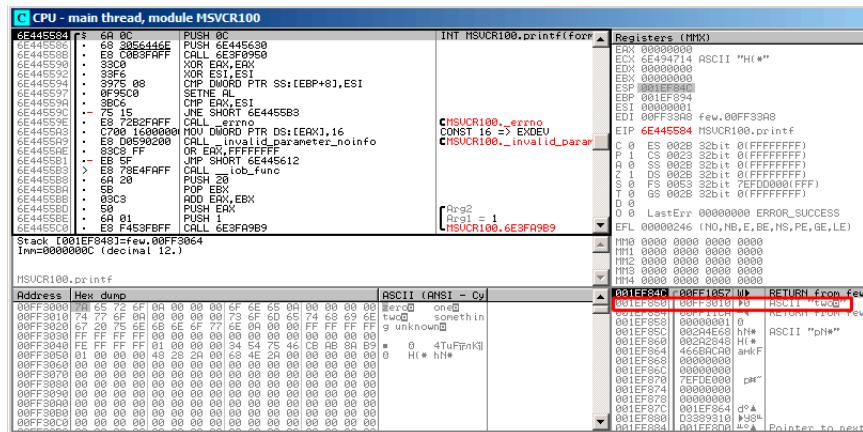


Figura 13.6: OllyDbg: printf() en MSVCR100.DLL

printf() 0x00FF3010 .

printf():

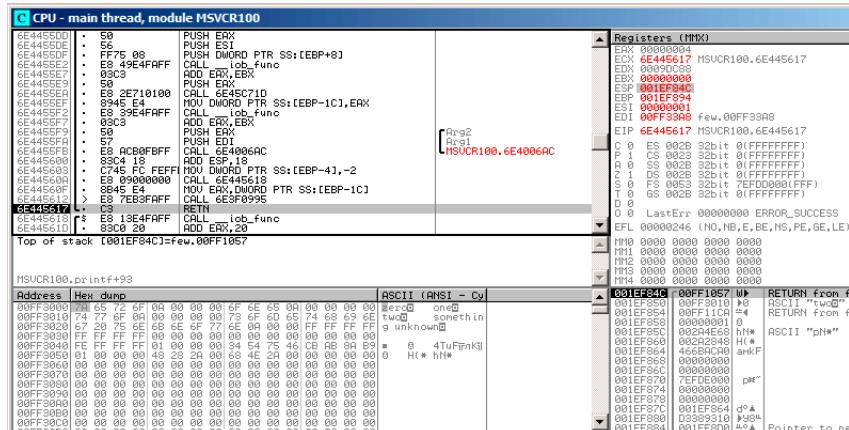


Figura 13.7: OllyDbg: printf() en MSVCR100.DLL

«tWO» .

F7 o F8 (pasar por encima) ... f() main():

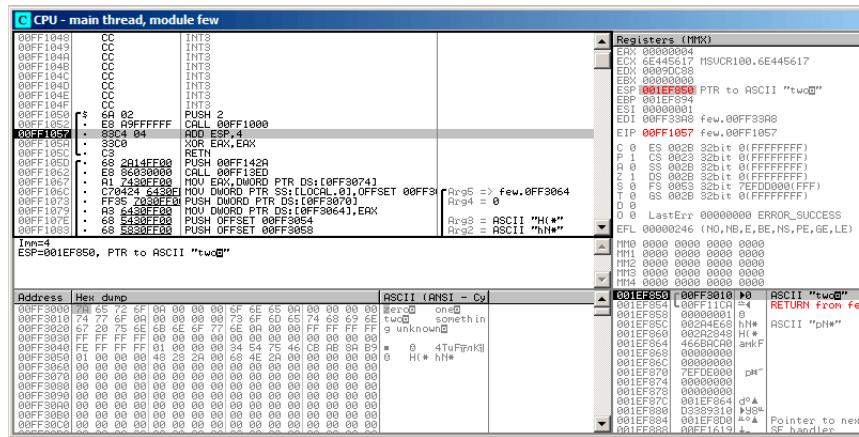


Figura 13.8: OllyDbg: main()

printf() main(). RA f() main(). CALL 0x00FF1000 f().

13.1.2 ARM: Con optimización Keil 6/2013 (Modo ARM)

.text:0000014C	f1:
.text:0000014C 00 00 50 E3	CMP R0, #0
.text:00000150 13 0E 8F 02	ADREQ R0, aZero ; "zero\n"
.text:00000154 05 00 00 0A	BEQ loc_170
.text:00000158 01 00 50 E3	CMP R0, #1
.text:0000015C 4B 0F 8F 02	ADREQ R0, aOne ; "one\n"
.text:00000160 02 00 00 0A	BEQ loc_170
.text:00000164 02 00 50 E3	CMP R0, #2
.text:00000168 4A 0F 8F 12	ADRNE R0, aSomethingUnknown ; "↙ something unknown\n"
.text:0000016C 4E 0F 8F 02	ADREQ R0, aTwo ; "two\n"
.text:00000170	loc_170: ; CODE XREF: f1+8
.text:00000170	; f1+14
.text:00000170 78 18 00 EA	B __2printf

BEQ loc_170, R0 = 0.

printf() (6.2.1 on page 60). printf().

CMP R0, #2 R0 = 2, «two\n».

13.1.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)

```

.text:000000D4          f1:
.text:000000D4 10 B5    PUSH   {R4,LR}
.text:000000D6 00 28    CMP    R0, #0
.text:000000D8 05 D0    BEQ    zero_case
.text:000000DA 01 28    CMP    R0, #1
.text:000000DC 05 D0    BEQ    one_case
.text:000000DE 02 28    CMP    R0, #2
.text:000000E0 05 D0    BEQ    two_case
.text:000000E2 91 A0    ADR    R0, aSomethingUnkno ; "↳
                        ↳ something unknown\n"
.text:000000E4 04 E0    B      default_case

.text:000000E6          zero_case: ; CODE XREF: f1+4
.text:000000E6 95 A0    ADR    R0, aZero ; "zero\n"
.text:000000E8 02 E0    B      default_case

.text:000000EA          one_case: ; CODE XREF: f1+8
.text:000000EA 96 A0    ADR    R0, aOne ; "one\n"
.text:000000EC 00 E0    B      default_case

.text:000000EE          two_case: ; CODE XREF: f1+C
.text:000000EE 97 A0    ADR    R0, aTwo ; "two\n"
.text:000000F0          default_case ; CODE XREF: f1+10
.text:000000F0           ; f1+14
.text:000000F0 06 F0 7E F8 BL     __2printf
.text:000000F4 10 BD    POP   {R4,PC}

```

13.1.4 ARM64: Sin optimización GCC (Linaro) 4.9

```

.LC12:
    .string "zero"
.LC13:
    .string "one"
.LC14:
    .string "two"
.LC15:
    .string "something unknown"
f12:
    stp    x29, x30, [sp, -32]!
    add    x29, sp, 0
    str    w0, [x29,28]
    ldr    w0, [x29,28]
    cmp    w0, 1
    beq    .L34

```

```

    cmp    w0, 2
    beq    .L35
    cmp    w0, wzr
    bne    .L38           ;
    adrp   x0, .LC12      ; "zero"
    add    x0, x0, :lo12:.LC12
    bl     puts
    b     .L32

.L34:
    adrp   x0, .LC13      ; "one"
    add    x0, x0, :lo12:.LC13
    bl     puts
    b     .L32

.L35:
    adrp   x0, .LC14      ; "two"
    add    x0, x0, :lo12:.LC14
    bl     puts
    b     .L32

.L38:
    adrp   x0, .LC15      ; "something unknown"
    add    x0, x0, :lo12:.LC15
    bl     puts
    nop

.L32:
    ldp    x29, x30, [sp], 32
    ret

```

«Hello, world!»: [3.4.5 on page 24](#).

13.1.5 ARM64: Con optimización GCC (Linaro) 4.9

```

f12:
    cmp    w0, 1
    beq    .L31
    cmp    w0, 2
    beq    .L32
    cbz    w0, .L35
;
    adrp   x0, .LC15      ; "something unknown"
    add    x0, x0, :lo12:.LC15
    b     puts

.L35:
    adrp   x0, .LC12      ; "zero"
    add    x0, x0, :lo12:.LC12
    b     puts

.L32:

```

```

    adrp    x0, .LC14      ; "two"
    add     x0, x0, :lo12:.LC14
    b      puts

.L31:
    adrp    x0, .LC13      ; "one"
    add     x0, x0, :lo12:.LC13
    b      puts

```

. CBZ (Compare and Branch on Zero) W0 . puts() 13.1.1 on page 184.

13.1.6 MIPS

Listing 13.3: Con optimización GCC 4.4.5 (IDA)

```

f:
; 1?
        lui    $gp, (__gnu_local_gp >> 16)
        li     $v0, 1
        beq   $a0, $v0, loc_60
        la    $gp, (__gnu_local_gp & 0xFFFF) ; branch
    ↳ delay slot
; 2?
        li     $v0, 2
        beq   $a0, $v0, loc_4C
        or    $at, $zero ; branch delay slot, NOP
; 0:
        bnez  $a0, loc_38
        or    $at, $zero ; branch delay slot, NOP
; :
        lui    $a0, ($LC0 >> 16) # "zero"
        lw     $t9, (puts & 0xFFFF)($gp)
        or    $at, $zero ; load delay slot, NOP
        jr    $t9 ; branch delay slot, NOP
        la    $a0, ($LC0 & 0xFFFF) # "zero" ; branch
    ↳ delay slot
# ↳
    ↳ -----
    ↳

loc_38:                      # CODE XREF: f+1C
        lui    $a0, ($LC3 >> 16) # "something unknown"
    ↳ "
        lw     $t9, (puts & 0xFFFF)($gp)
        or    $at, $zero ; load delay slot, NOP
        jr    $t9

```

```

        la      $a0, ($LC3 & 0xFFFF) # "something"
        ↵ unknown" ; branch delay slot
# ↵
        ↵ -----
        ↵

loc_4C:                                # CODE XREF: f+14
    lui    $a0, ($LC2 >> 16) # "two"
    lw     $t9, (puts & 0xFFFF)($gp)
    or     $at, $zero ; load delay slot, NOP
    jr     $t9
    la      $a0, ($LC2 & 0xFFFF) # "two" ; branch
        ↵ delay slot
# ↵
        ↵ -----
        ↵

loc_60:                                # CODE XREF: f+8
    lui    $a0, ($LC1 >> 16) # "one"
    lw     $t9, (puts & 0xFFFF)($gp)
    or     $at, $zero ; load delay slot, NOP
    jr     $t9
    la      $a0, ($LC1 & 0xFFFF) # "one" ; branch
        ↵ delay slot

```

: [13.1.1 on page 184](#).

«load delay slot»:..

13.1.7 Conclusión

Spanish text placeholder.[13.1.1](#).

13.2

```
#include <stdio.h>

void f (int a)
{
    switch (a)
    {
        case 0: printf ("zero\n"); break;
        case 1: printf ("one\n"); break;
        case 2: printf ("two\n"); break;
        case 3: printf ("three\n"); break;
        case 4: printf ("four\n"); break;
    }
}
```

```

    default: printf ("something unknown\n"); break;
}
};

int main()
{
    f (2); // test
};

```

13.2.1 x86

Sin optimización MSVC

(MSVC 2010):

Listing 13.4: MSVC 2010

```

tv64 = -4 ; size = 4
_a$ = 8 ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]
    mov     DWORD PTR tv64[ebp], eax
    cmp     DWORD PTR tv64[ebp], 4
    ja      SHORT $LN1@f
    mov     ecx, DWORD PTR tv64[ebp]
    jmp     DWORD PTR $LN11@f[ecx*4]
$LN6@f:
    push    OFFSET $SG739 ; 'zero', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN9@f
$LN5@f:
    push    OFFSET $SG741 ; 'one', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN9@f
$LN4@f:
    push    OFFSET $SG743 ; 'two', 0aH, 00H
    call    _printf
    add    esp, 4
    jmp     SHORT $LN9@f
$LN3@f:
    push    OFFSET $SG745 ; 'three', 0aH, 00H

```

```

call  _printf
add   esp, 4
jmp   SHORT $LN9@f
$LN2@f:
push  OFFSET $SG747 ; 'four', 0aH, 00H
call  _printf
add   esp, 4
jmp   SHORT $LN9@f
$LN1@f:
push  OFFSET $SG749 ; 'something unknown', 0aH, 00H
call  _printf
add   esp, 4
$LN9@f:
mov   esp, ebp
pop   ebp
ret   0
npad  2 ;
$LN11@f:
DD    $LN6@f ; 0
DD    $LN5@f ; 1
DD    $LN4@f ; 2
DD    $LN3@f ; 3
DD    $LN2@f ; 4
_f    ENDP

```

jumptable o *branch table*¹.

npad (88 on page 1141)

¹computed GOTO : [wikipedia](#). !

OllyDbg

OllyDbg. (2) EAX:

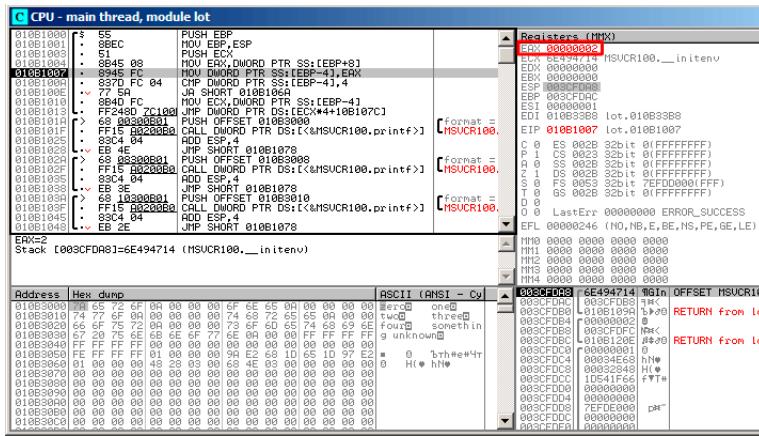


Figura 13.9: OllyDbg: EAX

4? :

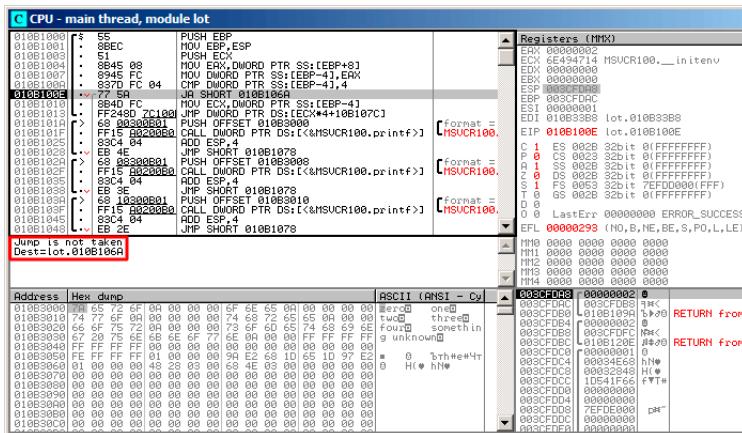


Figura 13.10: OllyDbg: 2 4:

jmpitable:

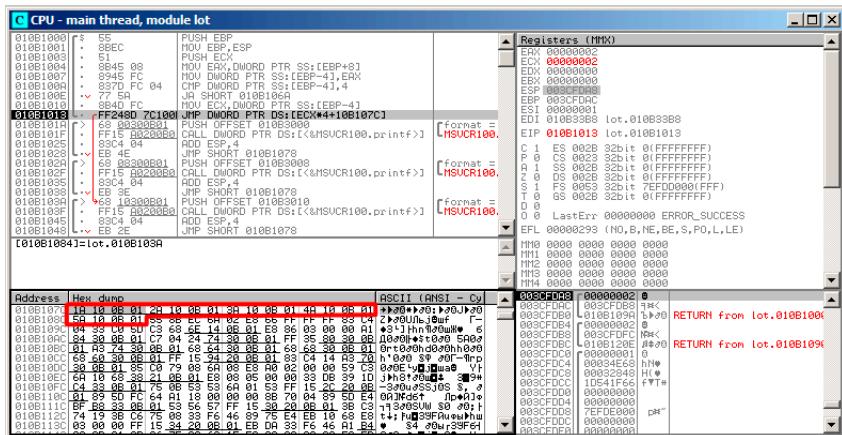


Figura 13.11: OllyDbg: jmpitable

². ECX 2. «Follow in Dump» → «Memory address» y OllyDbg. 0x010B103A.

²: 68.2.6 on page 903,

0x010B103A: «two»:

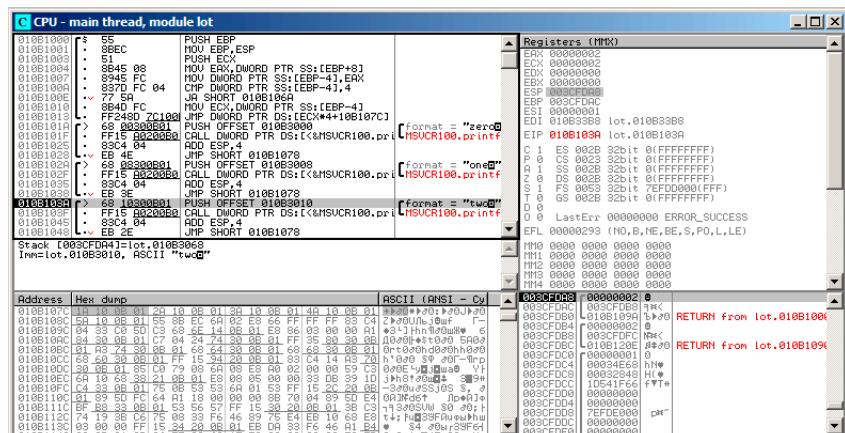


Figura 13.12: OllyDbg: case:

Sin optimización GCC

:

Listing 13.5: GCC 4.4.1

```

public f
f proc near ; CODE XREF: main+10

var_18 = dword ptr -18h
arg_0 = dword ptr 8

    push    ebp
    mov     ebp, esp
    sub     esp, 18h
    cmp     [ebp+arg_0], 4
    ja      short loc_8048444
    mov     eax, [ebp+arg_0]
    shl     eax, 2
    mov     eax, ds:off_804855C[eax]
    jmp     eax

loc_80483FE: ; DATA XREF: .rodata:off_804855C
    mov     [esp+18h+var_18], offset aZero ; "zero"
    call    _puts
    jmp     short locret_8048450

loc_804840C: ; DATA XREF: .rodata:08048560

```

```

    mov      [esp+18h+var_18], offset aOne ; "one"
    call     _puts
    jmp     short locret_8048450

loc_804841A: ; DATA XREF: .rodata:08048564
    mov      [esp+18h+var_18], offset aTwo ; "two"
    call     _puts
    jmp     short locret_8048450

loc_8048428: ; DATA XREF: .rodata:08048568
    mov      [esp+18h+var_18], offset aThree ; "three"
    call     _puts
    jmp     short locret_8048450

loc_8048436: ; DATA XREF: .rodata:0804856C
    mov      [esp+18h+var_18], offset aFour ; "four"
    call     _puts
    jmp     short locret_8048450

loc_8048444: ; CODE XREF: f+A
    mov      [esp+18h+var_18], offset aSomethingUnkno ; "↙
    ↳ something unknown"
    call     _puts

locret_8048450: ; CODE XREF: f+26
    ; f+34...
    leave
    retn
f      endp

off_804855C dd offset loc_80483FE      ; DATA XREF: f+12
            dd offset loc_804840C
            dd offset loc_804841A
            dd offset loc_8048428
            dd offset loc_8048436

```

13.2.2 ARM: Con optimización Keil 6/2013 (Modo ARM)

Listing 13.6: Con optimización Keil 6/2013 (Modo ARM)

00000174	f2				
00000174 05 00 50 E3	CMP	R0, #5			; switch 5 ↳
↳ cases					
00000178 00 F1 8F 30	ADCC	PC, PC, R0, LSL#2			; switch jump
0000017C 0E 00 00 EA	B	default_case			; jumptable ↳
↳ 00000178 default case					

```

00000180
00000180          loc_180 ; CODE XREF: f2+4
00000180 03 00 00 EA      B      zero_case      ; jumptable ↴
    ↳ 00000178 case 0

00000184
00000184          loc_184 ; CODE XREF: f2+4
00000184 04 00 00 EA      B      one_case      ; jumptable ↴
    ↳ 00000178 case 1

00000188
00000188          loc_188 ; CODE XREF: f2+4
00000188 05 00 00 EA      B      two_case      ; jumptable ↴
    ↳ 00000178 case 2

0000018C
0000018C          loc_18C ; CODE XREF: f2+4
0000018C 06 00 00 EA      B      three_case     ; jumptable ↴
    ↳ 00000178 case 3

00000190
00000190          loc_190 ; CODE XREF: f2+4
00000190 07 00 00 EA      B      four_case     ; jumptable ↴
    ↳ 00000178 case 4

00000194
00000194          zero_case ; CODE XREF: f2+4
00000194          ; f2:loc_180
00000194 EC 00 8F E2      ADR      R0, aZero      ; jumptable ↴
    ↳ 00000178 case 0
00000198 06 00 00 EA      B      loc_1B8

0000019C
0000019C          one_case ; CODE XREF: f2+4
0000019C          ; f2:loc_184
0000019C EC 00 8F E2      ADR      R0, aOne      ; jumptable ↴
    ↳ 00000178 case 1
000001A0 04 00 00 EA      B      loc_1B8

000001A4
000001A4          two_case ; CODE XREF: f2+4
000001A4          ; f2:loc_188
000001A4 01 0C 8F E2      ADR      R0, aTwo      ; jumptable ↴
    ↳ 00000178 case 2
000001A8 02 00 00 EA      B      loc_1B8

000001AC

```

```

000001AC      three_case ; CODE XREF: f2+4
000001AC          ; f2:loc_18C
000001AC 01 0C 8F E2    ADR      R0, aThree       ; jumptable ↴
    ↴ 00000178 case 3
000001B0 00 00 00 EA    B        loc_1B8

000001B4
000001B4      four_case ; CODE XREF: f2+4
000001B4          ; f2:loc_190
000001B4 01 0C 8F E2    ADR      R0, aFour       ; jumptable ↴
    ↴ 00000178 case 4
000001B8
000001B8      loc_1B8    ; CODE XREF: f2+24
000001B8          ; f2+2C
000001B8 66 18 00 EA    B        __2printf

000001BC
000001BC      default_case ; CODE XREF: f2+4
000001BC          ; f2+8
000001BC D4 00 8F E2    ADR      R0, aSomethingUnkno ; ↴
    ↴ jumptable 00000178 default case
000001C0 FC FF FF EA    B        loc_1B8

```

CMP R0, #5

ADDCC PC, PC, R0, LSL#2³ $R0 < 5$ (CC=Carry clear / Less than). ADDCC ($R0 \geq 5$), default_case.

$R0 < 5$ y ADDCC

. , LSL#2 (16.2.1 on page 267) «Shifts»,

$R0 * 4$,

ADDCC, (0x180) ADDCC (0x178),

$a = 1$, $PC + 8 + a * 4 = PC + 8 + 1 * 4 = PC + 12 = 0x184$.

switch(). Spanish text placeholder.

13.2.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)

Listing 13.7: Con optimización Keil 6/2013 (Modo Thumb)

000000F6	EXPORT f2
000000F6	f2
000000F6 10 B5	PUSH {R4,LR}
000000F8 03 00	MOVS R3,R0

³ADD

```

000000FA 06 F0 69 F8          BL      ↴
    ↳ __ARM_common_switch8_thumb ; switch 6 cases

000000FE 05                  DCB 5
000000FF 04 06 08 0A 0C 10    DCB 4, 6, 8, 0xA, 0xC, 0x10 ; ↵
    ↳ jump table for switch statement
00000105 00                  ALIGN 2
00000106
00000106                      zero_case ; CODE XREF: f2+4
00000106 8D A0                ADR     R0, aZero ; jumptable ↵
    ↳ 000000FA case 0
00000108 06 E0                B       loc_118

0000010A
0000010A                      one_case ; CODE XREF: f2+4
0000010A 8E A0                ADR     R0, aOne ; jumptable ↵
    ↳ 000000FA case 1
0000010C 04 E0                B       loc_118

0000010E
0000010E                      two_case ; CODE XREF: f2+4
0000010E 8F A0                ADR     R0, aTwo ; jumptable ↵
    ↳ 000000FA case 2
00000110 02 E0                B       loc_118

00000112
00000112                      three_case ; CODE XREF: f2+4
00000112 90 A0                ADR     R0, aThree ; jumptable ↵
    ↳ 000000FA case 3
00000114 00 E0                B       loc_118

00000116
00000116                      four_case ; CODE XREF: f2+4
00000116 91 A0                ADR     R0, aFour ; jumptable ↵
    ↳ 000000FA case 4
00000118
00000118                      loc_118 ; CODE XREF: f2+12
00000118                      ; f2+16
00000118 06 F0 6A F8          BL      __2printf
0000011C 10 BD                POP    {R4,PC}

0000011E
0000011E                      default_case ; CODE XREF: f2+4
0000011E 82 A0                ADR     R0, aSomethingUnkno ; ↵
    ↳ jumptable 000000FA default case
00000120 FA E7                B       loc_118

```

```

000061D0          EXPORT ↵
    ↳ __ARM_common_switch8_thumb
000061D0          __ARM_common_switch8_thumb ; CODE ↵
    ↳ XREF: example6_f2+4
000061D0 78 47      BX      PC

000061D2 00 00      ALIGN 4
000061D2          ; End of function ↵
    ↳ __ARM_common_switch8_thumb
000061D2
000061D4          _32__ARM_common_switch8_thumb ; ↵
    ↳ CODE XREF: __ARM_common_switch8_thumb
000061D4 01 C0 5E E5      LDRB    R12, [LR,#-1]
000061D8 0C 00 53 E1      CMP     R3, R12
000061DC 0C 30 DE 27      LDRCSB  R3, [LR,R12]
000061E0 03 30 DE 37      LDRCCB  R3, [LR,R3]
000061E4 83 C0 8E E0      ADD     R12, LR, R3,LSL#1
000061E8 1C FF 2F E1      BX      R12
000061E8          ; End of function ↵
    ↳ _32__ARM_common_switch8_thumb

```

__ARM_common_switch8_thumb. BX PC

BL __ARM_common_switch8_thumb

IDA

jumptable 000000FA case 0.

13.2.4 MIPS

Listing 13.8: Con optimización GCC 4.4.5 (IDA)

```

f:
    lui      $gp, (__gnu_local_gp >> 16)
;
    sltiu   $v0, $a0, 5
    bnez   $v0, loc_24
    la      $gp, (__gnu_local_gp & 0xFFFF) ; branch ↵
    ↳ delay slot
;
; "something unknown" :
    lui      $a0, ($LC5 >> 16) # "something unknown" ↵
    ↳ "
        lw      $t9, (puts & 0xFFFF)($gp)
        or      $at, $zero ; NOP
        jr      $t9

```

```

        la      $a0, ($LC5 & 0xFFFF) # "something" ↴
        ↳ unknown" ; branch delay slot

loc_24:                                # CODE XREF: f+8
;
;
;
; 4:                                la      $v0, off_120
; :                                sll     $a0, 2
; :                                addu   $a0, $v0, $a0
; :                                lw      $v0, 0($a0)
; :                                or      $at, $zero ; NOP
; :                                jr      $v0
; :                                or      $at, $zero ; branch delay slot, NOP

sub_44:                                # DATA XREF: .rodata ↴
        ↳ :0000012C
; "three"
        lui    $a0, ($LC3 >> 16) # "three"
        lw     $t9, (puts & 0xFFFF)($gp)
        or     $at, $zero ; NOP
        jr     $t9
        la     $a0, ($LC3 & 0xFFFF) # "three" ; ↴
        ↳ branch delay slot

sub_58:                                # DATA XREF: .rodata ↴
        ↳ :00000130
; "four"
        lui    $a0, ($LC4 >> 16) # "four"
        lw     $t9, (puts & 0xFFFF)($gp)
        or     $at, $zero ; NOP
        jr     $t9
        la     $a0, ($LC4 & 0xFFFF) # "four" ; branch ↴
        ↳ delay slot

sub_6C:                                # DATA XREF: .rodata: ↴
        ↳ off_120
; "zero"
        lui    $a0, ($LC0 >> 16) # "zero"
        lw     $t9, (puts & 0xFFFF)($gp)
        or     $at, $zero ; NOP
        jr     $t9
        la     $a0, ($LC0 & 0xFFFF) # "zero" ; branch ↴
        ↳ delay slot

```

```

sub_80:                                # DATA XREF: .rodata
    ↴ :00000124
;   "one"
        lui      $a0, ($LC1 >> 16)  # "one"
        lw       $t9, (puts & 0xFFFF)($gp)
        or       $at, $zero ; NOP
        jr       $t9
        la       $a0, ($LC1 & 0xFFFF)  # "one" ; branch
    ↴ delay slot

sub_94:                                # DATA XREF: .rodata
    ↴ :00000128
;   "two"
        lui      $a0, ($LC2 >> 16)  # "two"
        lw       $t9, (puts & 0xFFFF)($gp)
        or       $at, $zero ; NOP
        jr       $t9
        la       $a0, ($LC2 & 0xFFFF)  # "two" ; branch
    ↴ delay slot

; :
off_120:     .word sub_6C
              .word sub_80
              .word sub_94
              .word sub_44
              .word sub_58

```

BNEZ «Branch if Not Equal to Zero».

SLL («Shift Word Left Logical»).

13.2.5 Conclusión

switch():

Listing 13.9: x86

```

MOV REG, input
CMP REG, 4 ;
JA default
SHL REG, 2 ;
MOV REG, jump_table[REG]
JMP REG

case1:
;

```

```

        JMP exit
case2:
;
JMP exit
case3:
;
JMP exit
case4:
;
JMP exit
case5:
;
JMP exit

default:

    ...

exit:

    .....

jump_table dd case1
            dd case2
            dd case3
            dd case4
            dd case5

```

JMP jump_table[REG*4]. JMP jump_table[REG*8] en x64.

[18.5 on page 348](#).

13.3

```

#include <stdio.h>

void f(int a)
{
    switch (a)
    {
        case 1:
        case 2:
        case 7:
        case 10:
            printf ("1, 2, 7, 10\n");
            break;
    }
}

```

```

        case 3:
        case 4:
        case 5:
        case 6:
            printf ("3, 4, 5\n");
            break;
        case 8:
        case 9:
        case 20:
        case 21:
            printf ("8, 9, 21\n");
            break;
        case 22:
            printf ("22\n");
            break;
        default:
            printf ("default\n");
            break;
    };
}

int main()
{
    f(4);
}

```

13.3.1 MSVC

Listing 13.10: Con optimización MSVC 2010

```

1  $SG2798 DB      '1, 2, 7, 10', 0aH, 00H
2  $SG2800 DB      '3, 4, 5', 0aH, 00H
3  $SG2802 DB      '8, 9, 21', 0aH, 00H
4  $SG2804 DB      '22', 0aH, 00H
5  $SG2806 DB      'default', 0aH, 00H
6
7
8  _a$ = 8
9  _f      PROC
10   mov     eax, DWORD PTR _a$[esp-4]
11   dec     eax
12   cmp     eax, 21
13   ja      SHORT $LN1@f
14   movzx  eax, BYTE PTR $LN10@f[eax]
15   jmp     DWORD PTR $LN11@f[eax*4]
16 $LN5@f:

```

```

17    mov    DWORD PTR _a$[esp-4], OFFSET $SG2798 ; '1, 2, '
18    \ 7, 10' jmp    DWORD PTR __imp__printf
19 $LN4@f:
20    mov    DWORD PTR _a$[esp-4], OFFSET $SG2800 ; '3, 4, '
21    \ 5'   jmp    DWORD PTR __imp__printf
22 $LN3@f:
23    mov    DWORD PTR _a$[esp-4], OFFSET $SG2802 ; '8, 9, '
24    \ 21'  jmp    DWORD PTR __imp__printf
25 $LN2@f:
26    mov    DWORD PTR _a$[esp-4], OFFSET $SG2804 ; '22'
27    jmp    DWORD PTR __imp__printf
28 $LN1@f:
29    mov    DWORD PTR _a$[esp-4], OFFSET $SG2806 ; 'default'
30    \     jmp    DWORD PTR __imp__printf
31    npad  2 ; $LN11@f
32 $LN11@f:
33    DD    $LN5@f ; '1, 2, 7, 10'
34    DD    $LN4@f ; '3, 4, 5'
35    DD    $LN3@f ; '8, 9, 21'
36    DD    $LN2@f ; '22'
37    DD    $LN1@f ; 'default'
38 $LN10@f:
39    DB    0 ; a=1
40    DB    0 ; a=2
41    DB    1 ; a=3
42    DB    1 ; a=4
43    DB    1 ; a=5
44    DB    1 ; a=6
45    DB    0 ; a=7
46    DB    2 ; a=8
47    DB    2 ; a=9
48    DB    0 ; a=10
49    DB    4 ; a=11
50    DB    4 ; a=12
51    DB    4 ; a=13
52    DB    4 ; a=14
53    DB    4 ; a=15
54    DB    4 ; a=16
55    DB    4 ; a=17
56    DB    4 ; a=18
57    DB    4 ; a=19
58    DB    2 ; a=20
59    DB    2 ; a=21

```

```
60      DB      3 ; a=22
61      _f      ENDP
```

: (\$LN10@f), (\$LN11@f) .

(línea 13).

: 0 (1, 2, 7, 10), 1 (3, 4, 5), 2 (8, 9, 21), 3 (22), 4 .

(línea 14).

.

? ([13.2.1 on page 202](#)), ? .

13.3.2 GCC

GCC ([13.2.1 on page 202](#)), .

13.3.3 ARM64: Con optimización GCC 4.9.1

GCC 4.9.1 para ARM64 ...

Listing 13.11: Con optimización GCC 4.9.1 ARM64

```
f14:
;   W0
    sub    w0, w0, #1
    cmp    w0, 21
;
    bls    .L9
.L2:
; "default":
    adrp   x0, .LC4
    add    x0, x0, :lo12:.LC4
    b      puts
.L9:
; X1:
    adrp   x1, .L4
    add    x1, x1, :lo12:.L4
; W0=input_value-1
;
    ldrb   w0, [x1,w0,uxtw]
;
    adr    x1, .Lrtx4
;
    add    x0, x1, w0, sxtb #2
; :
```

```

    br      x0
;
.Lrtx4:
    .section      .rodata
;
.L4:
    .byte  (.L3 - .Lrtx4) / 4      ; case 1
    .byte  (.L3 - .Lrtx4) / 4      ; case 2
    .byte  (.L5 - .Lrtx4) / 4      ; case 3
    .byte  (.L5 - .Lrtx4) / 4      ; case 4
    .byte  (.L5 - .Lrtx4) / 4      ; case 5
    .byte  (.L5 - .Lrtx4) / 4      ; case 6
    .byte  (.L3 - .Lrtx4) / 4      ; case 7
    .byte  (.L6 - .Lrtx4) / 4      ; case 8
    .byte  (.L6 - .Lrtx4) / 4      ; case 9
    .byte  (.L3 - .Lrtx4) / 4      ; case 10
    .byte  (.L2 - .Lrtx4) / 4      ; case 11
    .byte  (.L2 - .Lrtx4) / 4      ; case 12
    .byte  (.L2 - .Lrtx4) / 4      ; case 13
    .byte  (.L2 - .Lrtx4) / 4      ; case 14
    .byte  (.L2 - .Lrtx4) / 4      ; case 15
    .byte  (.L2 - .Lrtx4) / 4      ; case 16
    .byte  (.L2 - .Lrtx4) / 4      ; case 17
    .byte  (.L2 - .Lrtx4) / 4      ; case 18
    .byte  (.L2 - .Lrtx4) / 4      ; case 19
    .byte  (.L6 - .Lrtx4) / 4      ; case 20
    .byte  (.L6 - .Lrtx4) / 4      ; case 21
    .byte  (.L7 - .Lrtx4) / 4      ; case 22
    .text
;
.L7:
; "22"
    adrp  x0, .LC3
    add   x0, x0, :lo12:.LC3
    b     puts
.L6:
; "8, 9, 21"
    adrp  x0, .LC2
    add   x0, x0, :lo12:.LC2
    b     puts
.L5:
; "3, 4, 5"
    adrp  x0, .LC1
    add   x0, x0, :lo12:.LC1
    b     puts
.L3:
; "1, 2, 7, 10"

```

```

        adrp    x0, .LC0
        add     x0, x0, :lo12:.LC0
        b      puts

.LC0:
        .string "1, 2, 7, 10"
.LC1:
        .string "3, 4, 5"
.LC2:
        .string "8, 9, 21"
.LC3:
        .string "22"
.LC4:
        .string "default"

```

Listing 13.12: jumptable in IDA

.rodata:00000000000000064	AREA .rodata, DATA,
↳ READONLY	
.rodata:00000000000000064	; ORG 0x64
.rodata:00000000000000064 \$d	DCB 9 ; case 1
.rodata:00000000000000065	DCB 9 ; case 2
.rodata:00000000000000066	DCB 6 ; case 3
.rodata:00000000000000067	DCB 6 ; case 4
.rodata:00000000000000068	DCB 6 ; case 5
.rodata:00000000000000069	DCB 6 ; case 6
.rodata:0000000000000006A	DCB 9 ; case 7
.rodata:0000000000000006B	DCB 3 ; case 8
.rodata:0000000000000006C	DCB 3 ; case 9
.rodata:0000000000000006D	DCB 9 ; case 10
.rodata:0000000000000006E	DCB 0xF7 ; case 11
.rodata:0000000000000006F	DCB 0xF7 ; case 12
.rodata:00000000000000070	DCB 0xF7 ; case 13
.rodata:00000000000000071	DCB 0xF7 ; case 14
.rodata:00000000000000072	DCB 0xF7 ; case 15
.rodata:00000000000000073	DCB 0xF7 ; case 16
.rodata:00000000000000074	DCB 0xF7 ; case 17
.rodata:00000000000000075	DCB 0xF7 ; case 18
.rodata:00000000000000076	DCB 0xF7 ; case 19
.rodata:00000000000000077	DCB 3 ; case 20
.rodata:00000000000000078	DCB 3 ; case 21
.rodata:00000000000000079	DCB 0 ; case 22
.rodata:0000000000000007B ; .rodata	ends

13.4 Fall-through

`switch() .:`

```

1 #define R 1
2 #define W 2
3 #define RW 3
4
5 void f(int type)
6 {
7     int read=0, write=0;
8
9     switch (type)
10    {
11        case RW:
12            read=1;
13        case W:
14            write=1;
15            break;
16        case R:
17            read=1;
18            break;
19        default:
20            break;
21    };
22    printf ("read=%d, write=%d\n", read, write);
23 }

```

type = 1 (R), read 1, type = 2 (W), write 2. type = 3 (RW), read y write 1.
type = RW type = W..

13.4.1 MSVC x86

Listing 13.13: MSVC 2012

```

$SG1305 DB      'read=%d, write=%d', 0aH, 00H

_write$ = -12    ; size = 4
_read$ = -8     ; size = 4
tv64 = -4       ; size = 4
_type$ = 8      ; size = 4
_f      PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 12
    mov     DWORD PTR _read$[ebp], 0
    mov     DWORD PTR _write$[ebp], 0
    mov     eax, DWORD PTR _type$[ebp]
    mov     DWORD PTR tv64[ebp], eax
    cmp     DWORD PTR tv64[ebp], 1 ; R

```

```

je      SHORT $LN2@f
cmp     DWORD PTR tv64[ebp], 2 ; W
je      SHORT $LN3@f
cmp     DWORD PTR tv64[ebp], 3 ; RW
je      SHORT $LN4@f
jmp     SHORT $LN5@f
$LN4@f: ; case RW:
    mov    DWORD PTR _read$[ebp], 1
$LN3@f: ; case W:
    mov    DWORD PTR _write$[ebp], 1
    jmp    SHORT $LN5@f
$LN2@f: ; case R:
    mov    DWORD PTR _read$[ebp], 1
$LN5@f: ; default
    mov    ecx, DWORD PTR _write$[ebp]
    push   ecx
    mov    edx, DWORD PTR _read$[ebp]
    push   edx
    push   OFFSET $SG1305 ; 'read=%d, write=%d'
    call   _printf
    add    esp, 12
    mov    esp, ebp
    pop    ebp
    ret    0
_f      ENDP

```

\$LN4@f y \$LN3@f: \$LN4@f, *read write.* type = W, \$LN3@f, .

13.4.2 ARM64

Listing 13.14: GCC (Linaro) 4.9

```

.LC0:
    .string "read=%d, write=%d\n"
f:
    stp    x29, x30, [sp, -48]!
    add    x29, sp, 0
    str    w0, [x29,28]
    str    wzr, [x29,44] ;
    str    wzr, [x29,40]
    ldr    w0, [x29,28] ;
    cmp    w0, 2           ; type=W?
    beq   .L3
    cmp    w0, 3           ; type=RW?
    beq   .L4
    cmp    w0, 1           ; type=R?

```

```
        beq    .L5
        b     .L6          ; ...
.L4: ; case RW
        mov    w0, 1
        str    w0, [x29,44] ; read=1
.L3: ; case W
        mov    w0, 1
        str    w0, [x29,40] ; write=1
        b     .L6
.L5: ; case R
        mov    w0, 1
        str    w0, [x29,44] ; read=1
        nop
.L6: ; default
        adrp   x0, .LC0 ; "read=%d, write=%d\n"
        add    x0, x0, :lo12:.LC0
        ldr    w1, [x29,44] ; "read"
        ldr    w2, [x29,40] ; "write"
        bl     printf
        ldp    x29, x30, [sp], 48
        ret
```

. .L4 y .L3.

13.5 Ejercicios

13.5.1 Ejercicio #1

: G.1.5 on page 1261.

Capítulo 14

Lazos

14.1

14.1.1 x86

Hay una instrucción especial LOOP en el conjunto de instrucciones x86 para verificar el valor en el registro ECX y si no es 0, **decrement** ECX y pasar el flujo de control a la etiqueta en el operando LOOP. Probablemente esta instrucción no es muy conveniente, y no hay compiladores modernos que la emitan automáticamente. Así que, si ves esta instrucción en algún lugar del código, es muy probable que sea un fragmento de código ensamblador escrito manualmente.

```
for().
```

```
{  
    ;  
}
```

```
:  
  
#include <stdio.h>  
  
void printing_function(int i)  
{  
    printf ("f(%d)\n", i);  
};
```

```

int main()
{
    int i;

    for (i=2; i<10; i++)
        printing_function(i);

    return 0;
}

```

(MSVC 2010):

Listing 14.1: MSVC 2010

```

_i$ = -4
_main    PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     DWORD PTR _i$[ebp], 2      ;
    jmp    SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp] ; :
    add     eax, 1                 ;
    mov     DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp     DWORD PTR _i$[ebp], 10   ;
    jge    SHORT $LN1@main         ;
    mov     ecx, DWORD PTR _i$[ebp] ; printing_function(i)
    push    ecx
    call    _printing_function
    add     esp, 4
    jmp    SHORT $LN2@main         ;
$LN1@main:
    xor     eax, eax
    mov     esp, ebp
    pop     ebp
    ret     0
_main    ENDP

```

Listing 14.2: GCC 4.4.1

```

main          proc near
var_20        = dword ptr -20h

```

```

var_4          = dword ptr -4

        push    ebp
        mov     ebp, esp
        and     esp, 0FFFFFFF0h
        sub     esp, 20h
        mov     [esp+20h+var_4], 2 ; 
        jmp     short loc_8048476

loc_8048465:
        mov     eax, [esp+20h+var_4]
        mov     [esp+20h+var_20], eax
        call    printing_function
        add     [esp+20h+var_4], 1 ; 

loc_8048476:
        cmp     [esp+20h+var_4], 9
        jle     short loc_8048465 ;
        mov     eax, 0
        leave
        retn
main      endp

```

(/0x):

Listing 14.3: Con optimización MSVC

```

_main    PROC
        push    esi
        mov     esi, 2
$LL3@main:
        push    esi
        call    _printing_function
        inc    esi
        add    esp, 4
        cmp    esi, 10      ; 0000000aH
        jl     SHORT $LL3@main
        xor    eax, eax
        pop    esi
        ret    0
_main    ENDP

```

Listing 14.4: Con optimización GCC 4.4.1

```

main      proc near
var_10    = dword ptr -10h

```

```

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 10h
mov     [esp+10h+var_10], 2
call    printing_function
mov     [esp+10h+var_10], 3
call    printing_function
mov     [esp+10h+var_10], 4
call    printing_function
mov     [esp+10h+var_10], 5
call    printing_function
mov     [esp+10h+var_10], 6
call    printing_function
mov     [esp+10h+var_10], 7
call    printing_function
mov     [esp+10h+var_10], 8
call    printing_function
mov     [esp+10h+var_10], 9
call    printing_function
xor    eax, eax
leave
retn
endp
main

```

1.

Listing 14.5: GCC

```

public main
proc near
var_20      = dword ptr -20h

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
push    ebx
mov     ebx, 2      ; i=2
sub     esp, 1Ch

; :
nop

```

1: [Dre07]. : [Int14, pág. 3.4.1.7].

```
loc_80484D0:  
; :  
    mov    [esp+20h+var_20], ebx  
    add    ebx, 1      ; i++  
    call   printing_function  
    cmp    ebx, 64h   ; i==100?  
    jnz    short loc_80484D0 ;  
    add    esp, 1Ch  
    xor    eax, eax  ; 0  
    pop    ebx  
    mov    esp, ebp  
    pop    ebp  
    retn  
main    endp
```

14.1.2 x86: OllyDbg

MSVC 2010 /0x y /0b0 OllyDbg.

OllyDbg:

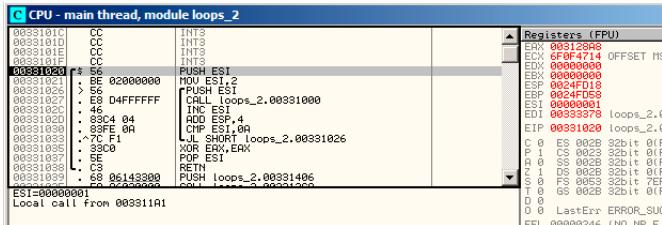


Figura 14.1: OllyDbg:

(F8 – pasar por encima) ESI $ESI = i = 6$:

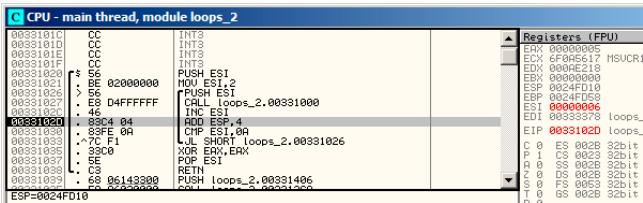


Figura 14.2: OllyDbg: $i = 6$

9 . JL

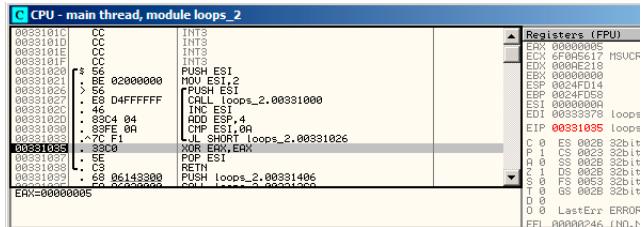


Figura 14.3: OllyDbg: $ESI = 10$,

14.1.3 x86: tracer

```
tracer.exe -l:loops_2.exe bpx=loops_2.exe!0x00401026
```

BPX

tracer.log :

```
PID=12884|New process loops_2.exe
(0) loops_2.exe!0x401026
EAX=0x000a328c8 EBX=0x000000000 ECX=0x6f0f4714 EDX=0x000000000
ESI=0x00000002 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=PF ZF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000003 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF PF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000004 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF PF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000005 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000006 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF PF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000007 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000008 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF AF SF IF
(0) loops_2.exe!0x401026
EAX=0x000000005 EBX=0x000000000 ECX=0x6f0a5617 EDX=0x000ee188
ESI=0x00000009 EDI=0x00333378 EBP=0x0024fbfc ESP=0x0024fb8
EIP=0x00331026
FLAGS=CF PF AF SF IF
PID=12884|Process loops_2.exe exited. ExitCode=0 (0x0)
```

ESI

```
trace.. IDA main() 0x00401020 :
```

```
tracer.exe -l:loops_2.exe bpf=loops_2.exe!0x00401020,trace:cc
```

BPF.

loops_2.exe.idc y loops_2.exe_clear.idc.

loops_2.exe.idc IDA:

```
.text:00401020 ; ====== S U B R O U T I N E ======
.text:00401020
.text:00401020
.text:00401020
.text:00401020 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401020 _main proc near ; CODE XREF: __tmainCRTStartup+11Dpp
.text:00401020
.text:00401020
.text:00401020     argc      = dword ptr  4
.text:00401020     argv      = dword ptr  8
.text:00401020     envp      = dword ptr  0Ch
.text:00401020
.text:00401020     push    esi          ; ESI=1
.text:00401021     mov     esi, 2
.text:00401022
.text:00401026 loc_401026:      ; CODE XREF: _main+13ij
.text:00401026     push    esi          ; ESI=2..9
.text:00401027     call    sub_401000 ; tracing nested maximum level (1) reached,
.text:00401028     inc     esi          ; ESI=2..9
.text:00401029     add     esp, 4       ; ESP=0x38fcbc
.text:00401030     cmp     esi, 0Ah        ; ESI=3..0xa
.text:00401031     jl      short loc_401026 ; SF=false,true OF=false
.text:00401032     xor     eax, eax
.text:00401033     pop     esi
.text:00401034     retn    ; EAX=0
.text:00401038 main  endo
```

Figura 14.4: IDA

ESI. main() EAX.

tracer loops_2.exe.txt,:>

Listing 14.6: loops_2.exe.txt

```
0x401020 (.text+0x20), e=      1 [PUSH ESI] ESI=1
0x401021 (.text+0x21), e=      1 [MOV ESI, 2]
0x401026 (.text+0x26), e=      8 [PUSH ESI] ESI=2..9
0x401027 (.text+0x27), e=      8 [CALL 8D1000h] tracing nested?
    ↳ maximum level (1) reached, skipping this CALL 8D1000h=0x2
    ↳ x8d1000
0x40102c (.text+0x2c), e=      8 [INC ESI] ESI=2..9
0x40102d (.text+0x2d), e=      8 [ADD ESP, 4] ESP=0x38fcfc
0x401030 (.text+0x30), e=      8 [CMP ESI, 0Ah] ESI=3..0xa
0x401033 (.text+0x33), e=      8 [JL 8D1026h] SF=false,true OF?
    ↳ =false
0x401035 (.text+0x35), e=      1 [XOR EAX, EAX]
0x401037 (.text+0x37), e=      1 [POP ESI]
0x401038 (.text+0x38), e=      1 [RETN] EAX=0
```

14.1.4 ARM

Sin optimización Keil 6/2013 (Modo ARM)

```

main
    STMFD   SP!, {R4,LR}
    MOV     R4, #2
    B      loc_368
loc_35C ; CODE XREF: main+1C
    MOV     R0, R4
    BL     printing_function
    ADD     R4, R4, #1

loc_368 ; CODE XREF: main+8
    CMP     R4, #0xA
    BLT    loc_35C
    MOV     R0, #0
    LDMFD  SP!, {R4,PC}

```

«MOV R4, #2» *i.*
 «MOV R0, R4» y «BL printing_function» ,
 «ADD R4, R4, #1»
 «CMP R4, #0xA» *i* 0xA (10).

Con optimización Keil 6/2013 (Modo Thumb)

```

_main
    PUSH    {R4,LR}
    MOVS    R4, #2

loc_132
    MOVS    R0, R4
    BL     printing_function
    ADDS    R4, R4, #1
    CMP     R4, #0xA
    BLT    loc_132
    MOVS    R0, #0
    POP     {R4,PC}

```

Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

_main
    PUSH    {R4,R7,LR}
    MOVW    R4, #0x1124 ; "%d\n"
    MOVS    R1, #2
    MOVT.W R4, #0
    ADD    R7, SP, #4

```

```

ADD      R4, PC
MOV      R0, R4
BLX      _printf
MOV      R0, R4
MOVS     R1, #3
BLX      _printf
MOV      R0, R4
MOVS     R1, #4
BLX      _printf
MOV      R0, R4
MOVS     R1, #5
BLX      _printf
MOV      R0, R4
MOVS     R1, #6
BLX      _printf
MOV      R0, R4
MOVS     R1, #7
BLX      _printf
MOV      R0, R4
MOVS     R1, #8
BLX      _printf
MOV      R0, R4
MOVS     R1, #9
BLX      _printf
MOVS     R0, #0
POP      {R4,R7,PC}

```

```

void printing_function(int i)
{
    printf ("%d\n", i);
}

```

LLVM,..

ARM64: Con optimización GCC 4.9.1

Listing 14.7: Con optimización GCC 4.9.1

```

printing_function:
; printf():
    mov      w1, w0
;
    adrp    x0, .LC0
    add    x0, x0, :lo12:.LC0
; :
    b       printf

```

```

main:
; :
    stp      x29, x30, [sp, -32]!
; :
    add      x29, sp, 0
; :
    str      x19, [sp,16]
; .
; :
    mov      w19, 2
.L3:
; printing_function():
    mov      w0, w19
; .
    add      w19, w19, 1
; .
    bl       printing_function
; ?
    cmp      w19, 10
; :
    bne      .L3
; 0
    mov      w0, 0
; :
    ldr      x19, [sp,16]
; :
    ldp      x29, x30, [sp], 32
    ret
.LC0:
.string "f(%d)\n"

```

ARM64: Sin optimización GCC 4.9.1

Listing 14.8: Sin optimización GCC 4.9.1 -fno-inline

```

printing_function:
; printf():
    mov      w1, w0
;
    adrp    x0, .LC0
    add      x0, x0, :lo12:.LC0
; :
    b       printf
main:
; :
    stp      x29, x30, [sp, -32]!

```

```

; :
;       add      x29,  sp,  0
; :
;       str      x19,  [sp,16]
; :
;       mov      w19,  2
.L3:
;   printing_function():
;       mov      w0,  w19
;   :
;       add      w19,  w19,  1
;   :
;       bl      printing_function
;   ?
;       cmp      w19,  10
;   :
;       bne      .L3
;   0
;       mov      w0,  0
;   :
;       ldr      x19,  [sp,16]
;   :
;       ldp      x29,  x30,  [sp],  32
;       ret
.LC0:
       .string "f(%d)\n"

```

14.1.5 MIPS

Listing 14.9: Sin optimización GCC 4.4.5 (IDA)

```

main:
;
;
i           = -0x10
saved_FP    = -8
saved_RA    = -4

;
addiu     $sp, -0x28
sw        $ra, 0x28+saved_RA($sp)
sw        $fp, 0x28+saved_FP($sp)
move     $fp, $sp
;
```

```

        li      $v0, 2
        sw      $v0, 0x28+i($fp)
; . "BEQ $ZERO, $ZERO, loc_9C":
        b       loc_9C
        or      $at, $zero ; branch delay slot, NOP
# ↵
    ↵ ----- ↵
    ↵

loc_80:                                # CODE XREF: main+48
; printing_function():
        lw      $a0, 0x28+i($fp)
        jal     printing_function
        or      $at, $zero ; branch delay slot, NOP
; :
        lw      $v0, 0x28+i($fp)
        or      $at, $zero ; NOP
        addiu   $v0, 1
        sw      $v0, 0x28+i($fp)

loc_9C:                                # CODE XREF: main+18
;
        lw      $v0, 0x28+i($fp)
        or      $at, $zero ; NOP
        slti   $v0, 0xA
; :
        bnez   $v0, loc_80
        or      $at, $zero ; branch delay slot, NOP
; :
        move   $v0, $zero
; :
        move   $sp, $fp
        lw      $ra, 0x28+saved_RA($sp)
        lw      $fp, 0x28+saved_FP($sp)
        addiu   $sp, 0x28
        jr      $ra
        or      $at, $zero ; branch delay slot, NOP

```

«B». (BEQ).

14.1.6

: i

```

for (i=0; i<total_entries_to_process; i++)
;
```

total_entries_to_process 0, .

14.2

```
#include <stdio.h>

void my_memcpy (unsigned char* dst, unsigned char* src, size_t ↴
    ↴ cnt)
{
    size_t i;
    for (i=0; i<cnt; i++)
        dst[i]=src[i];
}
```

14.2.1

Listing 14.10: GCC 4.9 x64 (-Os)

```
my_memcpy:
; RDI =
; RSI =
; RDX =
;
        xor     eax, eax
.L2:
;
        cmp     rax, rdx
        je      .L5
; RSI+i:
        mov     cl, BYTE PTR [rsi+rax]
; RDI+i:
        mov     BYTE PTR [rdi+rax], cl
        inc     rax ; i++
        jmp     .L2
.L5:
        ret
```

Listing 14.11: GCC 4.9 ARM64 (-Os)

```
my_memcpy:
; X0 =
; X1 =
```

```

; X2 =
;
        mov      x3, 0
.L2:
;
        cmp      x3, x2
        beq      .L5
; X1+i:
        ldrb    w4, [x1,x3]
; X1+i:
        strb    w4, [x0,x3]
        add     x3, x3, 1 ; i++
        b       .L2
.L5:
        ret

```

Listing 14.12: Con optimización Keil 6/2013 (Modo Thumb)

```

my_memcpy PROC
; R0 =
; R1 =
; R2 =
;
        PUSH    {r4,lr}
;
        MOVS   r3,#0
;
        B      |L0.12|
|L0.6|
; R1+i:
        LDRB   r4,[r1,r3]
; R1+i:
        STRB   r4,[r0,r3]
; i++
        ADDS   r3,r3,#1
|L0.12|
; i<size?
        CMP    r3,r2
;
        BCC   |L0.6|
        POP    {r4,pc}
        ENDP

```

14.2.2 ARM

Listing 14.13: Con optimización Keil 6/2013 (Modo ARM)

```

my_memcpy PROC
; R0 =
; R1 =
; R2 =
;
        MOV      r3,#0
|L0.4|
;
        CMP      r3,r2
;
; R2<R3  i<size.
; R1+i:
        LDRBCC  r12,[r1,r3]
; R1+i:
        STRBCC  r12,[r0,r3]
; i++
        ADDCC   r3,r3,#1
;
; i<size
; ( i>=size)
        BCC     |L0.4|
;
        BX      lr
ENDP

```

14.2.3 MIPS

Listing 14.14: GCC 4.4.5 (-Os) (IDA)

```

my_memcpy:
;
        b      loc_14
;
; \$v0:
        move   $v0, $zero ; branch delay slot
loc_8:                                # CODE XREF: my_memcpy
    ↳ +1C
;
        lbu    $v1, 0($t0)
;
; (i):
        addiu $v0, 1

```

```

; $a3
        sb      $v1, 0($a3)

loc_14:                                # CODE XREF: my_memcpy
;
        sltu    $v1, $v0, $a2
;
        addu    $t0, $a1, $v0
; $t0 = $a1+$v0 = src+i
; "cnt":
        bnez    $v1, loc_8
; ($a3 = $a0+$v0 = dst+i):
        addu    $a3, $a0, $v0 ; branch delay slot
; BNEZ
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP

```

LBU («Load Byte Unsigned») y SB («Store Byte»).

14.2.4

Con optimización GCC : [25.1.2 on page 540](#).

14.3 Conclusión

:

Listing 14.15: x86

```

        mov [counter], 2 ;
        jmp check
body:
;
;
;
add [counter], 1 ;
check:
        cmp [counter], 9
        jle body

```

Listing 14.16: x86

```

MOV [counter], 2 ;
JMP check

```

```
body:  
;  
;  
;  
    MOV REG, [counter] ;  
    INC REG  
    MOV [counter], REG  
check:  
    CMP [counter], 9  
    JLE body
```

Listing 14.17: x86

```
MOV EBX, 2 ;  
JMP check  
body:  
;  
;  
;  
    INC EBX ;  
check:  
    CMP EBX, 9  
    JLE body
```

Listing 14.18: x86

```
MOV [counter], 2 ;  
JMP label_check  
label_increment:  
    ADD [counter], 1 ;  
label_check:  
    CMP [counter], 10  
    JGE exit  
    ;  
    ;  
    ;  
    JMP label_increment  
exit:
```

Listing 14.19: x86

```
MOV REG, 2 ;  
body:  
;
```

```
;  
;  
INC REG ;  
CMP REG, 10  
JL body
```

Listing 14.20: x86

```
;  
MOV ECX, 10  
body:  
;  
;  
;  
;  
LOOP body
```

ARM.

Listing 14.21: ARM

```
MOV R4, 2 ;  
B check  
body:  
;  
;  
;  
ADD R4,R4, #1 ;  
check:  
CMP R4, #10  
BLT body
```

14.4 Ejercicios

14.4.1 Ejercicio #1

LOOP?

14.4.2 Ejercicio #2

(14.1.1 on page 219), OS [6..20].

14.4.3 Ejercicio #3

Lo que hace el código?

Listing 14.22: Con optimización MSVC 2010

```
$SG2795 DB      '%d', 0aH, 00H

_main PROC
    push    esi
    push    edi
    mov     edi, DWORD PTR __imp__printf
    mov     esi, 100
    npad   3 ; align next label
$LL3@main:
    push    esi
    push    OFFSET $SG2795 ; '%d'
    call    edi
    dec    esi
    add    esp, 8
    test   esi, esi
    jg     SHORT $LL3@main
    pop    edi
    xor    eax, eax
    pop    esi
    ret    0
_main ENDP
```

Listing 14.23: Sin optimización Keil 6/2013 (Modo ARM)

```
main PROC
    PUSH   {r4,lr}
    MOV    r4,#0x64
|L0.8|
    MOV    r1,r4
    ADR    r0,|L0.40|
    BL     _2printf
    SUB    r4,r4,#1
    CMP    r4,#0
    MOVLE r0,#0
    BGT    |L0.8|
    POP    {r4,pc}
    ENDP

|L0.40|
    DCB    "%d\n",0
```

Listing 14.24: Sin optimización Keil 6/2013 (Modo Thumb)

```

main PROC
    PUSH    {r4,lr}
    MOVS    r4,#0x64
|L0.4|
    MOVS    r1,r4
    ADR     r0,|L0.24|
    BL      __2printf
    SUBS   r4,r4,#1
    CMP    r4,#0
    BGT    |L0.4|
    MOVS   r0,#0
    POP    {r4,pc}
    ENDP

    DCW    0x0000
|L0.24|
    DCB    "%d\n",0

```

Listing 14.25: Con optimización GCC 4.9 (ARM64)

```

main:
    stp    x29, x30, [sp, -32]!
    add    x29, sp, 0
    stp    x19, x20, [sp,16]
    adrp   x20, .LC0
    mov    w19, 100
    add    x20, x20, :lo12:.LC0
.L2:
    mov    w1, w19
    mov    x0, x20
    bl     printf
    subs   w19, w19, #1
    bne   .L2
    ldp    x19, x20, [sp,16]
    ldp    x29, x30, [sp], 32
    ret
.LC0:
    .string "%d\n"

```

Listing 14.26: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

main:
var_18      = -0x18
var_C       = -0xC
var_8       = -8
var_4       = -4

```

```

        lui      $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x28
        la      $gp, (__gnu_local_gp & 0xFFFF)
        sw      $ra, 0x28+var_4($sp)
        sw      $s1, 0x28+var_8($sp)
        sw      $s0, 0x28+var_C($sp)
        sw      $gp, 0x28+var_18($sp)
        la      $s1, $LC0      # "%d\n"
        li      $s0, 0x64      # 'd'

loc_28:                                # CODE XREF: main+40
        lw      $t9, (printf & 0xFFFF)($gp)
        move   $a1, $s0
        move   $a0, $s1
        jalr   $t9
        addiu $s0, -1
        lw      $gp, 0x28+var_18($sp)
        bnez   $s0, loc_28
        or      $at, $zero
        lw      $ra, 0x28+var_4($sp)
        lw      $s1, 0x28+var_8($sp)
        lw      $s0, 0x28+var_C($sp)
        jr      $ra
        addiu $sp, 0x28

$LC0:         .ascii "%d\n" <0>          # DATA XREF: main+1C

```

: G.1.7 on page 1261.

14.4.4 Ejercicio #4

Lo que hace el código?

Listing 14.27: Con optimización MSVC 2010

```

$SG2795 DB      '%d', 0aH, 00H

_main  PROC
        push   esi
        push   edi
        mov    edi, DWORD PTR __imp__printf
        mov    esi, 1
        npad  3 ; align next label
$LL3@main:
        push   esi
        push   OFFSET $SG2795 ; '%d'
        call   edi

```

```

add    esi, 3
add    esp, 8
cmp    esi, 100
j1    SHORT $LL3@main
pop    edi
xor    eax, eax
pop    esi
ret    0
_main  ENDP

```

Listing 14.28: Sin optimización Keil 6/2013 (Modo ARM)

```

main PROC
    PUSH   {r4,lr}
    MOV    r4,#1
|L0.8|
    MOV    r1,r4
    ADR    r0,|L0.40|
    BL     __2printf
    ADD    r4,r4,#3
    CMP    r4,#0x64
    MOVGE r0,#0
    BLT    |L0.8|
    POP    {r4,pc}
    ENDP

|L0.40|
    DCB    "%d\n",0

```

Listing 14.29: Sin optimización Keil 6/2013 (Modo Thumb)

```

main PROC
    PUSH   {r4,lr}
    MOVS  r4,#1
|L0.4|
    MOVS  r1,r4
    ADR   r0,|L0.24|
    BL    __2printf
    ADDS r4,r4,#3
    CMP   r4,#0x64
    BLT  |L0.4|
    MOVS r0,#0
    POP   {r4,pc}
    ENDP

    DCW   0x0000
|L0.24|
    DCB   "%d\n",0

```

Listing 14.30: Con optimización GCC 4.9 (ARM64)

```

main:
    stp    x29, x30, [sp, -32]!
    add    x29, sp, 0
    stp    x19, x20, [sp,16]
    adrp   x20, .LC0
    mov    w19, 1
    add    x20, x20, :lo12:.LC0
.L2:
    mov    w1, w19
    mov    x0, x20
    add    w19, w19, 3
    bl     printf
    cmp    w19, 100
    bne   .L2
    ldp   x19, x20, [sp,16]
    ldp   x29, x30, [sp], 32
    ret
.LC0:
.string "%d\n"

```

Listing 14.31: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

main:
var_18      = -0x18
var_10      = -0x10
var_C       = -0xC
var_8       = -8
var_4       = -4

        lui    $gp, (__gnu_local_gp >> 16)
        addiu $sp, -0x28
        la    $gp, (__gnu_local_gp & 0xFFFF)
        sw    $ra, 0x28+var_4($sp)
        sw    $s2, 0x28+var_8($sp)
        sw    $s1, 0x28+var_C($sp)
        sw    $s0, 0x28+var_10($sp)
        sw    $gp, 0x28+var_18($sp)
        la    $s2, $LC0      # "%d\n"
        li    $s0, 1
        li    $s1, 0x64      # 'd'

loc_30:          # CODE XREF: main+48
        lw    $t9, (printf & 0xFFFF)($gp)
        move $a1, $s0
        move $a0, $s2

```

```
jalr    $t9
addiu   $s0, 3
lw      $gp, 0x28+var_18($sp)
bne    $s0, $s1, loc_30
or     $at, $zero
lw      $ra, 0x28+var_4($sp)
lw      $s2, 0x28+var_8($sp)
lw      $s1, 0x28+var_C($sp)
lw      $s0, 0x28+var_10($sp)
jr     $ra
addiu   $sp, 0x28

$LCO: .ascii "%d\n"<0> # DATA XREF: main+20
```

: G.1.8 on page 1261.

Capítulo 15

Procesamiento de strings C simples

15.1 `strlen()`

```
int my_strlen (const char * str)
{
    const char *eos = str;

    while( *eos++ ) ;

    return( eos - str - 1 );
}

int main()
{
    // test
    return my_strlen("hello!");
}
```

15.1.1 x86

Sin optimización MSVC

```
_eos$ = -4          ; size = 4
_str$ = 8          ; size = 4
_strlen PROC
```

```

push    ebp
mov     ebp, esp
push    ecx
mov     eax, DWORD PTR _str$[ebp] ; "str"
mov     DWORD PTR _eos$[ebp], eax ; "eos"
$LN2@strlen_:
    mov    ecx, DWORD PTR _eos$[ebp] ; ECX=eos
;

movsx   edx, BYTE PTR [ecx]
mov     eax, DWORD PTR _eos$[ebp] ; EAX=eos
add    eax, 1                  ; EAX
mov     DWORD PTR _eos$[ebp], eax ; "eos"
test   edx, edx               ; EDX ?
je     SHORT $LN1@strlen_
jmp    SHORT $LN2@strlen_
$LN1@strlen_:
;

mov     eax, DWORD PTR _eos$[ebp]
sub    eax, DWORD PTR _str$[ebp]
sub    eax, 1                  ;
mov     esp, ebp
pop    ebp
ret    0
_strlen_ ENDP

```

[«» \(30 on page 584\).](#)

Sin optimización GCC

GCC 4.4.1:

```

strlen      public strlen
strlen      proc near

eos         = dword ptr -4
arg_0       = dword ptr  8

push    ebp
mov     ebp, esp
sub    esp, 10h
mov     eax, [ebp+arg_0]
mov     [ebp+eos], eax

```

```

loc_80483F0:
    mov     eax, [ebp+eos]
    movzx  eax, byte ptr [eax]
    test   al, al
    setnz  al
    add    [ebp+eos], 1
    test   al, al
    jnz    short loc_80483F0
    mov    edx, [ebp+eos]
    mov    eax, [ebp+arg_0]
    mov    ecx, edx
    sub    ecx, eax
    mov    eax, ecx
    sub    eax, 1
    leave
    retn
strlen  endp

```

Con optimización MSVC

:

Listing 15.1: Con optimización MSVC 2012 /Ob0

```

__str$ = 8                      ; size = 4
_strlen PROC
    mov    edx, DWORD PTR __str$[esp-4] ; EDX ->
    mov    eax, edx                   ; EAX
$LL2@strlen:
    mov    cl, BYTE PTR [eax]        ; CL = *EAX
    inc    eax                     ; EAX++
    test   cl, cl                  ; CL==0?
    jne    SHORT $LL2@strlen      ;
    sub    eax, edx                ;
    dec    eax                     ; EAX
    ret    0
_strlen ENDP

```

INC/DEC

Con optimización MSVC + OllyDbg

OllyDbg. :

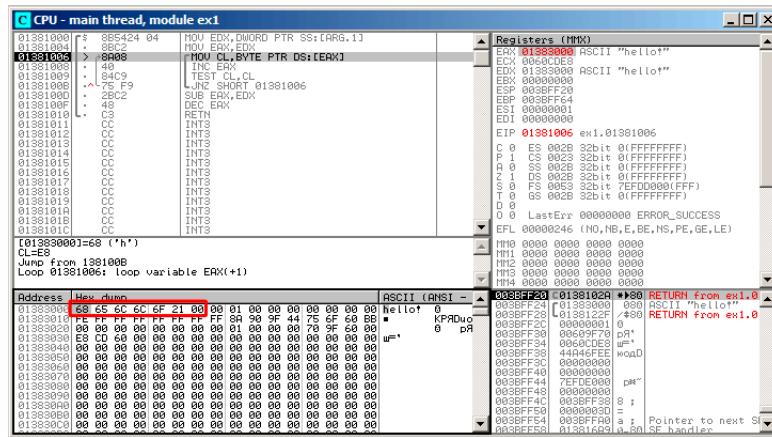


Figura 15.1: OllyDbg:

EAX, «Follow in Dump» «hello!». .

F8 (pasar por encima) :

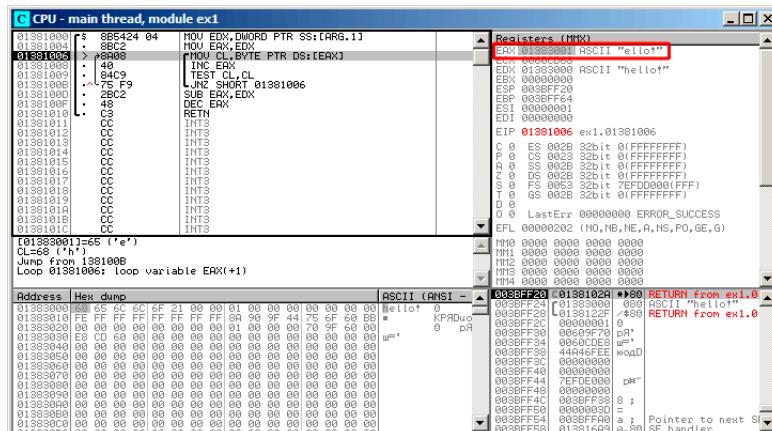


Figura 15.2: OllyDbg:

EAX

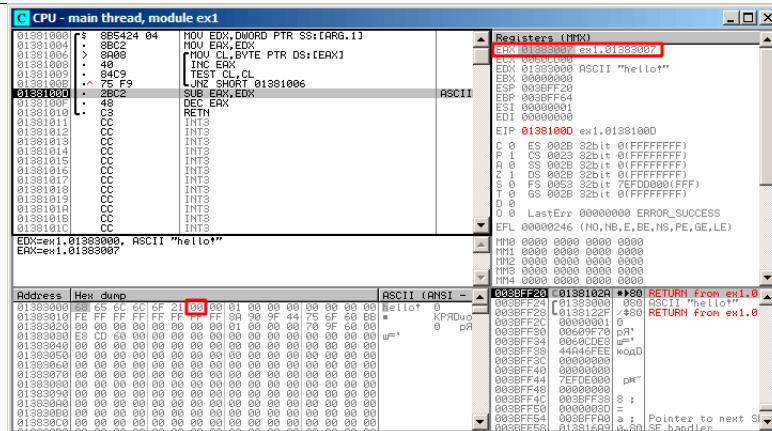


Figura 15.3: OllyDbg:

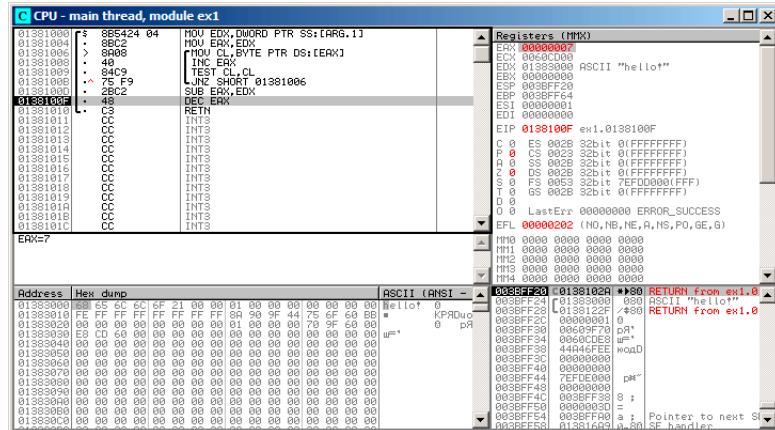


Figura 15.4: OllyDbg:

«hello!», 7. `strlen()`

Con optimización GCC

```

public strlen
strlen proc near

arg_0      = dword ptr  8

push    ebp
mov     ebp, esp
mov     ecx, [ebp+arg_0]
mov     eax, ecx

loc_8048418:
    movzx  edx, byte ptr [eax]
    add    eax, 1
    test   dl, dl
    jnz    short loc_8048418
    not    ecx
    add    eax, ecx
    pop    ebp
    retn
strlen endp

mov dl, byte ptr [eax].

```

«» (30 on page 584).

```
ecx=str;
eax=eos;
ecx=(-ecx)-1;
eax=eax+ecx
return eax
```

...:

```
ecx=str;
eax=eos;
eax=eax-ecx;
eax=eax-1;
return eax
```

15.1.2 ARM

32- ARM

Sin optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Listing 15.2: Sin optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```
_strlen

eos  = -8
str  = -4

SUB   SP, SP, #8 ;
STR   R0, [SP,#8+str]
LDR   R0, [SP,#8+str]
STR   R0, [SP,#8+eos]

loc_2CB8 ; CODE XREF: _strlen+28
    LDR   R0, [SP,#8+eos]
    ADD   R1, R0, #1
    STR   R1, [SP,#8+eos]
    LDRSB R0, [R0]
    CMP   R0, #0
    BEQ   loc_2CD4
    B     loc_2CB8
loc_2CD4 ; CODE XREF: _strlen+24
    LDR   R0, [SP,#8+eos]
    LDR   R1, [SP,#8+str]
```

```

SUB    R0, R0, R1 ; R0=eos-str
SUB    R0, R0, #1 ; R0=R0-1
ADD    SP, SP, #8 ;
BX    LR

```

eos y str.

str y eos.

loc_2CB8.

(LDR, ADD, STR) *eos R0.*

LDRSB R0, [R0] («Load Register Signed Byte»)¹. MOVSX x86. ([15.1.1 on page 246](#)).

, LDRSB CMP y BEQ

eos y str

N.B. .

Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

Listing 15.3: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

```

_strlen
        MOV      R1, R0

loc_2DF6
        LDRB.W  R2, [R1],#1
        CMP      R2, #0
        BNE    loc_2DF6
        MVNS    R0, R0
        ADD     R0, R1
        BX     LR

```

LDRB.W R2, [R1],#1

: [28.2 on page 574](#).

MVNS² *eos - str - 1. R0 = str + eos,* ([15.1.1 on page 251](#)).

¹.

²MoVe Not

Con optimización Keil 6/2013 (Modo ARM)

Listing 15.4: Con optimización Keil 6/2013 (Modo ARM)

```
_strlen
        MOV      R1, R0
loc_2C8
        LDRB    R2, [R1], #1
        CMP     R2, #0
        SUBEQ   R0, R1, R0
        SUBEQ   R0, R0, #1
        BNE    loc_2C8
        BX     LR
```

str - eos - 1 -EQ , , SUBEQ

ARM64**Con optimización GCC (Linaro) 4.9**

```
my_strlen:
        mov      x1, x0
        ; X1
.L58:
        ;
        ldrb    w2, [x1], 1
        ; Compare and Branch if NonZero:
        cbnz   w2, .L58
        ;
        sub     x0, x1, x0
        ;
        sub     w0, w0, #1
        ret
```

[15.1.1 on page 247](#)...

Sin optimización GCC (Linaro) 4.9

```
my_strlen:
;
        sub     sp, sp, #32
```

```

; [sp,8]
    str      x0,  [sp,8]
    ldr      x0,  [sp,8]
;
    str      x0,  [sp,24]
    nop
.L62:
; eos++
    ldr      x0,  [sp,24] ; "eos" X0
    add      x1, x0, 1     ; X0
    str      x1,  [sp,24] ; X0 "eos"
;
    ldrb    w0,  [x0]
;
    cmp      w0, wzr
; (Branch Not Equal)
    bne      .L62
;
; "eos" X1
    ldr      x1,  [sp,24]
; "str" X0
    ldr      x0,  [sp,8]
;
    sub      x0, x1, x0
;
    sub      w0, w0, #1
;
    add      sp, sp, 32
    ret

```

...

15.1.3 MIPS

Listing 15.5: Con optimización GCC 4.4.5 (IDA)

```

my_strlen:
; $v1:
    move    $v1, $a0

loc_4:
; $a1:
    lb      $a1, 0($v1)
    or      $at, $zero ; load delay slot, NOP
; loc_4:
    bnez   $a1, loc_4

```

```
; :
        addiu   $v1, 1 ; branch delay slot
; :
        nor     $v0, $zero, $a0
; $v0=-str-1
        jr      $ra
; = $v1 + $v0 = eos + ( -str-1 ) = eos - str - 1
        addu   $v0, $v1, $v0 ; branch delay slot
```

NOR DST, \$ZERO, SRC.

15.2 Ejercicios

15.2.1 Ejercicio #1

Lo que hace el código?

Listing 15.6: Con optimización MSVC 2010

```
_s$ = 8
_f    PROC
        mov     edx, DWORD PTR _s$[esp-4]
        mov     cl, BYTE PTR [edx]
        xor     eax, eax
        test   cl, cl
        je     SHORT $LN2@f
        npad  4 ; align next label
$LL4@f:
        cmp     cl, 32
        jne     SHORT $LN3@f
        inc     eax
$LN3@f:
        mov     cl, BYTE PTR [edx+1]
        inc     edx
        test   cl, cl
        jne     SHORT $LL4@f
$LN2@f:
        ret     0
_f    ENDP
```

Listing 15.7: GCC 4.8.1 -O3

```
f:
.LFB24:
        push   ebx
```

```

    mov      ecx, DWORD PTR [esp+8]
    xor      eax, eax
    movzx   edx, BYTE PTR [ecx]
    test    dl, dl
    je      .L2
.L3:
    cmp      dl, 32
    lea      ebx, [eax+1]
    cmove   eax, ebx
    add     ecx, 1
    movzx   edx, BYTE PTR [ecx]
    test    dl, dl
    jne     .L3
.L2:
    pop     ebx
    ret

```

Listing 15.8: Con optimización Keil 6/2013 (Modo ARM)

```

f PROC
    MOV      r1,#0
|L0.4|
    LDRB    r2,[r0,#0]
    CMP      r2,#0
    MOVEQ   r0,r1
    BXEQ    lr
    CMP      r2,#0x20
    ADDEQ   r1,r1,#1
    ADD     r0,r0,#1
    B       |L0.4|
    ENDP

```

Listing 15.9: Con optimización Keil 6/2013 (Modo Thumb)

```

f PROC
    MOVS   r1,#0
    B      |L0.12|
|L0.4|
    CMP    r2,#0x20
    BNE   |L0.10|
    ADDS  r1,r1,#1
|L0.10|
    ADDS  r0,r0,#1
|L0.12|
    LDRB  r2,[r0,#0]
    CMP   r2,#0
    BNE   |L0.4|
    MOVS  r0,r1

```

```
BX      lr
ENDP
```

Listing 15.10: Con optimización GCC 4.9 (ARM64)

```
f:
    ldrb    w1, [x0]
    cbz    w1, .L4
    mov    w2, 0
.L3:
    cmp    w1, 32
    ldrb    w1, [x0,1]!
    csinc   w2, w2, w2, ne
    cbnz    w1, .L3
.L2:
    mov    w0, w2
    ret
.L4:
    mov    w2, w1
    b     .L2
```

Listing 15.11: Con optimización GCC 4.4.5 (MIPS) (IDA)

```
f:
    lb      $v1, 0($a0)
    or      $at, $zero
    beqz   $v1, locret_48
    li      $a1, 0x20 # ' '
    b       loc_28
    move   $v0, $zero
loc_18:                                # CODE XREF: f:loc_28
    lb      $v1, 0($a0)
    or      $at, $zero
    beqz   $v1, locret_40
    or      $at, $zero
loc_28:                                # CODE XREF: f+10
                                         # f+38
    bne   $v1, $a1, loc_18
    addiu $a0, 1
    lb      $v1, 0($a0)
    or      $at, $zero
    bnez   $v1, loc_28
    addiu $v0, 1
locret_40:                               # CODE XREF: f+20
    jr      $ra
    or      $at, $zero
```

locret_48:	# CODE XREF: f+8
jr \$ra	
move \$v0, \$zero	

Responda [G.1.9 on page 1262](#).

Capítulo 16

Substitución de instrucciones aritméticas por otros

, ADD y SUB: línea 18 en Spanish text placeholder.[52.1.](#)

[A.6.2 on page 1233.](#)

16.1

16.1.1

:

Listing 16.1: Con optimización MSVC 2010

```
unsigned int f(unsigned int a)
{
    return a*8;
};
```

```
_TEXT    SEGMENT
_a$ = 8
        ↓ = 4                                     ; size ↴
_f      PROC
; File c:\polygon\c\2.c
    mov     eax, DWORD PTR _a$[esp-4]
    add     eax, eax
    add     eax, eax
    add     eax, eax
```

```

        ret      0
_f      ENDP
_TEXT   ENDS
END

```

16.1.2

```

unsigned int f(unsigned int a)
{
    return a*4;
}

```

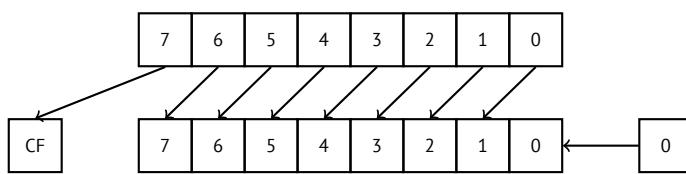
Listing 16.2: Sin optimización MSVC 2010

```

_a$ = 8          ; size = 4
_f      PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    shl     eax, 2
    pop     ebp
    ret     0
_f      ENDP

```

:



ARM:

Listing 16.3: Sin optimización Keil 6/2013 (Modo ARM)

```

f      PROC
    LSL      r0,r0,#2
    BX      lr
    ENDP

```

MIPS:

Listing 16.4: Con optimización GCC 4.4.5 (IDA)

jr	\$ra
sll	\$v0, \$a0, 2 ; branch delay slot

SLL «Shift Left Logical».

16.1.3

32-

```
#include <stdint.h>

int f1(int a)
{
    return a*7;
};

int f2(int a)
{
    return a*28;
};

int f3(int a)
{
    return a*17;
};
```

x86

Listing 16.5: Con optimización MSVC 2012

```
; a*7
_a$ = 8
_f1 PROC
    mov    ecx, DWORD PTR _a$[esp-4]
; ECX=a
    lea    eax, DWORD PTR [ecx*8]
; EAX=ECX*8
    sub    eax, ecx
; EAX=EAX-ECX=ECX*8-ECX=ECX*7=a*7
    ret    0
_f1 ENDP
; a*28
```

```

_a$ = 8
_f2    PROC
        mov      ecx, DWORD PTR _a$[esp-4]
; ECX=a
        lea      eax, DWORD PTR [ecx*8]
; EAX=ECX*8
        sub      eax, ecx
; EAX=EAX-ECX=ECX*8-ECX=ECX*7=a*7
        shl      eax, 2
; EAX=EAX<<2=(a*7)*4=a*28
        ret      0
_f2    ENDP

; a*17
_a$ = 8
_f3    PROC
        mov      eax, DWORD PTR _a$[esp-4]
; EAX=a
        shl      eax, 4
; EAX=EAX<<4=EAX*16=a*16
        add      eax, DWORD PTR _a$[esp-4]
; EAX=EAX+a=a*16+a=a*17
        ret      0
_f3    ENDP

```

ARM

Listing 16.6: Con optimización Keil 6/2013 (Modo ARM)

```

; a*7
||f1|| PROC
        RSB      r0,r0,r0,LSL #3
; R0=R0<<3-R0=R0*8-R0=a*8-a=a*7
        BX      lr
        ENDP

; a*28
||f2|| PROC
        RSB      r0,r0,r0,LSL #3
; R0=R0<<3-R0=R0*8-R0=a*8-a=a*7
        LSL      r0,r0,#2
; R0=R0<<2=R0*4=a*7*4=a*28
        BX      lr
        ENDP

; a*17

```

```
||f3|| PROC
    ADD      r0,r0,r0,LSL #4
; R0=R0+R0<<4=R0+R0*16=R0*17=a*17
    BX      lr
ENDP
```

:

Listing 16.7: Con optimización Keil 6/2013 (Modo Thumb)

```
; a*7
||f1|| PROC
    LSLS      r1,r0,#3
; R1=R0<<3=a<<3=a*8
    SUBS      r0,r1,r0
; R0=R1-R0=a*8-a=a*7
    BX      lr
ENDP

; a*28
||f2|| PROC
    MOVS      r1,#0x1c ; 28
; R1=28
    MULS      r0,r1,r0
; R0=R1*R0=28*a
    BX      lr
ENDP

; a*17
||f3|| PROC
    LSLS      r1,r0,#4
; R1=R0<<4=R0*16=a*16
    ADDS      r0,r0,r1
; R0=R0+R1=a+a*16=a*17
    BX      lr
ENDP
```

MIPS

Listing 16.8: Con optimización GCC 4.4.5 (IDA)

```
_f1:
        sll      $v0, $a0, 3
; $v0 = $a0<<3 = $a0*8
        jr      $ra
        subu   $v0, $a0 ; branch delay slot
```

```

; $v0 = $v0-$a0 = $a0*8-$a0 = $a0*7

_f2:
        sll      $v0, $a0, 5
; $v0 = $a0<<5 = $a0*32
        sll      $a0, 2
; $a0 = $a0<<2 = $a0*4
        jr      $ra
        subu    $v0, $a0 ; branch delay slot
; $v0 = $a0*32-$a0*4 = $a0*28

_f3:
        sll      $v0, $a0, 4
; $v0 = $a0<<4 = $a0*16
        jr      $ra
        addu    $v0, $a0 ; branch delay slot
; $v0 = $a0*16+$a0 = $a0*17

```

64-

```

#include <stdint.h>

int64_t f1(int64_t a)
{
    return a*7;
};

int64_t f2(int64_t a)
{
    return a*28;
};

int64_t f3(int64_t a)
{
    return a*17;
};

```

x64

Listing 16.9: Con optimización MSVC 2012

```

; a*7
f1:
        lea      rax, [0+rdi*8]
; RAX=RDI*8=a*8

```

```

        sub      rax, rdi
; RAX=RAX-RDI=a*8-a=a*7
        ret

; a*28
f2:
        lea      rax, [0+rdi*4]
; RAX=RDIX4=a*4
        sal     rdi, 5
; RDI=RDIX5=RDIX32=a*32
        sub     rdi, rax
; RDI=RDIX32-a*4=a*28
        mov      rax, rdi
        ret

; a*17
f3:
        mov      rax, rdi
        sal     rax, 4
; RAX=RAWX4=a*16
        add     rax, rdi
; RAX=a*16+a=a*17
        ret

```

ARM64

Listing 16.10: Con optimización GCC (Linaro) 4.9 ARM64

```

; a*7
f1:
        lsl      x1, x0, 3
; X1=X0<<3=X0*8=a*8
        sub     x0, x1, x0
; X0=X1-X0=a*8-a=a*7
        ret

; a*28
f2:
        lsl      x1, x0, 5
; X1=X0<<5=a*32
        sub     x0, x1, x0, lsl 2
; X0=X1-X0<<2=a*32-a<<2=a*32-a*4=a*28
        ret

; a*17
f3:

```

```

add      x0, x0, x0, lsl 4
; X0=X0+X0<<4=a+a*16=a*17
    ret

```

16.2

16.2.1

:

```

unsigned int f(unsigned int a)
{
    return a/4;
}

```

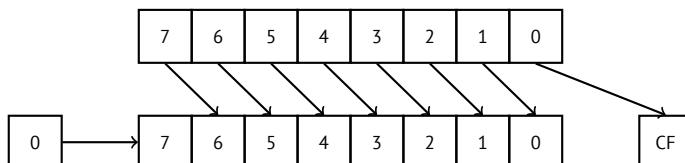
(MSVC 2010):

Listing 16.11: MSVC 2010

```

_a$ = 8                                     ; size 4
    \ = 4
_f     PROC
    mov     eax, DWORD PTR _a$[esp-4]
    shr     eax, 2
    ret     0
_f     ENDP

```



Es fácil de entender si imaginas el número 23 en el sistema numérico decimal. 23 se puede dividir fácilmente por 10 simplemente eliminando el último dígito (3 – resto de la división). 2 queda después de la operación como **quotient**.

ARM:

Listing 16.12: Sin optimización Keil 6/2013 (Modo ARM)

```

f PROC
    LSR     r0,r0,#2

```

```
BX      lr
ENDP
```

MIPS:

Listing 16.13: Con optimización GCC 4.4.5 (IDA)

```
jr      $ra
srl     $v0, $a0, 2 ; branch delay slot
```

«Shift Right Logical».

16.3 Ejercicios

16.3.1 Ejercicio #2

Lo que hace el código?

Listing 16.14: Con optimización MSVC 2010

```
_a$ = 8
_f PROC
    mov     ecx, DWORD PTR _a$[esp-4]
    lea     eax, DWORD PTR [ecx*8]
    sub     eax, ecx
    ret     0
_f ENDP
```

Listing 16.15: Sin optimización Keil 6/2013 (Modo ARM)

```
f PROC
    RSB     r0,r0,r0,LSL #3
    BX      lr
ENDP
```

Listing 16.16: Sin optimización Keil 6/2013 (Modo Thumb)

```
f PROC
    LSLS   r1,r0,#3
    SUBS   r0,r1,r0
    BX     lr
ENDP
```

Listing 16.17: Con optimización GCC 4.9 (ARM64)

```
f:
    lsl     w1, w0, 3
```

```
sub      w0, w1, w0  
ret
```

Listing 16.18: Con optimización GCC 4.4.5 (MIPS) (IDA)

```
f:  
    sll      $v0, $a0, 3  
    jr      $ra  
    subu    $v0, $a0
```

Responda [G.1.10 on page 1262](#).

Capítulo 17

Unidad de Punto flotante

17.1 IEEE 754

17.2 x86

[1](#).

.

17.3 ARM, MIPS, x86/x64 SIMD

17.4 C/C++

17.5

:

```
#include <stdio.h>

double f (double a, double b)
{
    return a/3.14 + b*4.1;
};

int main()
{
```

```
    printf ("%f\n", f(1.2, 3.4));
};
```

17.5.1 x86

MSVC

MSVC 2010:

Listing 17.1: MSVC 2010: f()

```
CONST SEGMENT
__real@4010666666666666 DQ 0401066666666666r ; 4.1
CONST ENDS
CONST SEGMENT
__real@40091eb851eb851f DQ 040091eb851eb851fr ; 3.14
CONST ENDS
_TEXT SEGMENT
_a$ = 8           ; size = 8
_b$ = 16          ; size = 8
_f PROC
    push    ebp
    mov     ebp, esp
    fld     QWORD PTR _a$[ebp]

; : ST(0) = _a
    fdiv   QWORD PTR __real@40091eb851eb851f
; : ST(0) =
    fld     QWORD PTR _b$[ebp]
; : ST(0) = _b; ST(1) =
    fmul   QWORD PTR __real@4010666666666666
; :
; ST(0) = ;
; ST(1) =
    faddp  ST(1), ST(0)
; : ST(0) =
    pop    ebp
```

ret	0
_f	ENDP

MSVC + OllyDbg

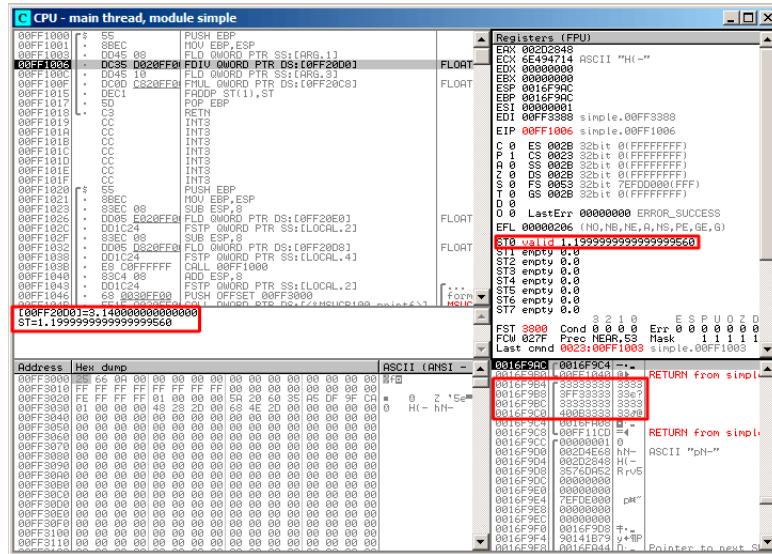


Figura 17.1: OllyDbg:

, OllyDbg

. FDIV, ST(0) (quotient):

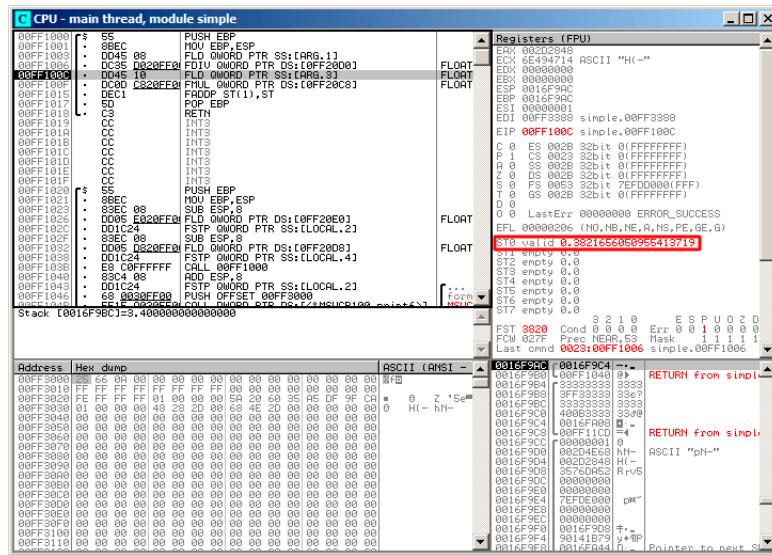


Figura 17.2: OllyDbg: FDIW

: FLD :

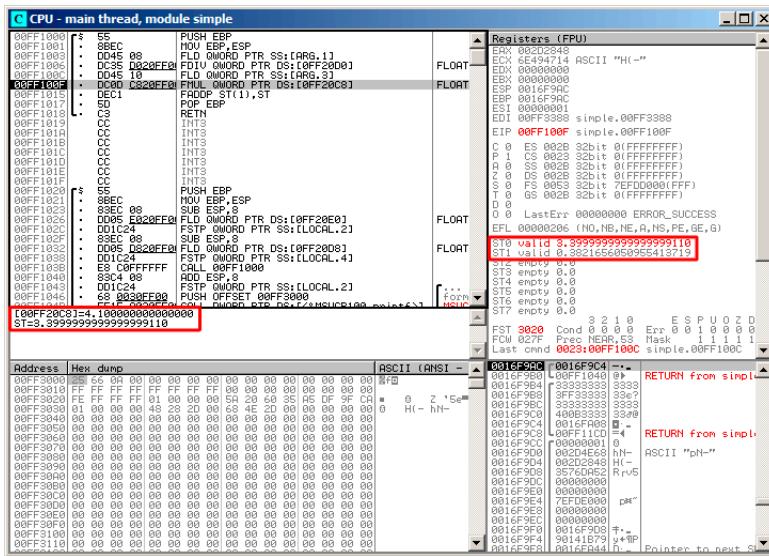


Figura 17.3: OllyDbg:

quotient ST(1).:FMUL.

:FMUL:

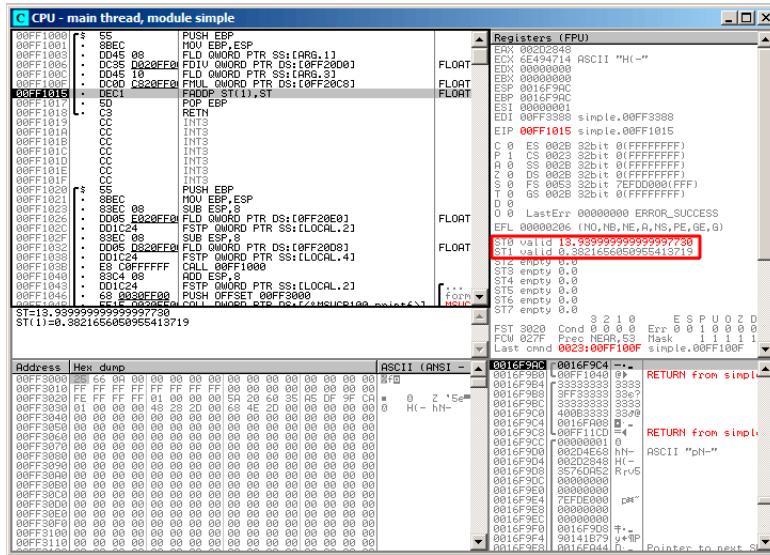


Figura 17.4: OllyDbg: FMUL

: FADDP:

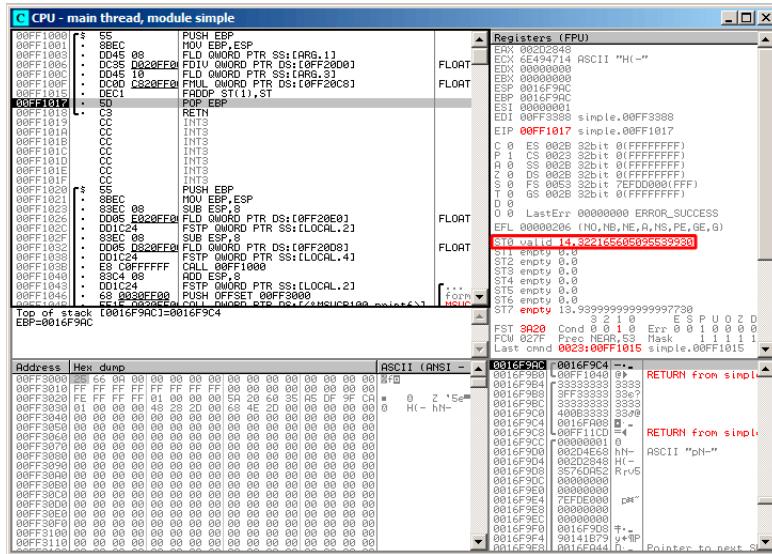


Figura 17.5: OllyDbg: FADDP

?

:17.2 on page 270..

(...)

GCC

Listing 17.2: Con optimización GCC 4.4.1

```
        public f
        proc near

arg_0      = qword ptr  8
arg_8      = qword ptr  10h

        push    ebp
        fld     ds:dbl_8048608 ; 3.14

; : ST(0) = 3.14

        mov     ebp, esp
        fdivr [ebp+arg_0]
```

```

; : ST(0) =
          fld      ds:dbl_8048610 ; 4.1

; : ST(0) = 4.1, ST(1) =
          fmul    [ebp+arg_8]

; : ST(0) = , ST(1) =
          pop     ebp
          faddp   st(1), st

; : ST(0) =
          retn
f           endp

```

17.5.2 ARM: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

,. VFP (*Vector Floating Point*).

Listing 17.3: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```

f
          VLDR      D16, =3.14
          VMOV      D17, R0, R1 ; "a"
          VMOV      D18, R2, R3 ; "b"
          VDIV.F64 D16, D17, D16 ; a/3.14
          VLDR      D17, =4.1
          VMUL.F64 D17, D18, D17 ; b*4.1
          VADD.F64 D16, D17, D16 ; +
          VMOV      R0, R1, D16
          BX        LR

dbl_2C98  DCFD 3.14          ; DATA XREF: f
dbl_2CA0  DCFD 4.1          ; DATA XREF: f+10

```

: B.3.3 on page 1248.

VLDR y VMOV

VMOV D17, R0, R1 D17.

VMOV R0, R1, D16 : D16 R0 y R1, ,

VDIV, VMUL y VADD,

17.5.3 ARM: Con optimización Keil 6/2013 (Modo Thumb)

```

f
    PUSH    {R3-R7,LR}
    MOVS    R7, R2
    MOVS    R4, R3
    MOVS    R5, R0
    MOVS    R6, R1
    LDR     R2, =0x66666666 ; 4.1
    LDR     R3, =0x40106666
    MOVS    R0, R7
    MOVS    R1, R4
    BL      __aeabi_dmul
    MOVS    R7, R0
    MOVS    R4, R1
    LDR     R2, =0x51EB851F ; 3.14
    LDR     R3, =0x40091EB8
    MOVS    R0, R5
    MOVS    R1, R6
    BL      __aeabi_ddiv
    MOVS    R2, R7
    MOVS    R3, R4
    BL      __aeabi_dadd
    POP    {R3-R7,PC}

; 4.1 :
dword_364    DCD 0x66666666          ; DATA XREF: f+A
dword_368    DCD 0x40106666          ; DATA XREF: f+C
; 3.14 :
dword_36C    DCD 0x51EB851F          ; DATA XREF: f+1A
dword_370    DCD 0x40091EB8          ; DATA XREF: f+1C

```

__aeabi_dmul, __aeabi_ddiv, __aeabi_dadd

17.5.4 ARM64: Con optimización GCC (Linaro) 4.9

:

Listing 17.4: Con optimización GCC (Linaro) 4.9

```

f:
; D0 = a, D1 = b
        ldr     d2, .LC25      ; 3.14
; D2 = 3.14
        fdiv   d0, d0, d2
; D0 = D0/D2 = a/3.14

```

```

        ldr      d2, .LC26      ; 4.1
; D2 = 4.1
        fmadd   d0, d1, d2, d0
; D0 = D1*D2+D0 = b*4.1+a/3.14
        ret

; :
.LC25:
.word   1374389535      ; 3.14
.word   1074339512

.LC26:
.word   1717986918      ; 4.1
.word   1074816614

```

17.5.5 ARM64: Sin optimización GCC (Linaro) 4.9

Listing 17.5: Sin optimización GCC (Linaro) 4.9

```

f:
        sub    sp, sp, #16
        str    d0, [sp,8]      ; "a" Register Save Area
        str    d1, [sp]         ; "b" Register Save Area
        ldr    x1, [sp,8]
; X1 = a
        ldr    x0, .LC25
; X0 = 3.14
        fmov   d0, x1
        fmov   d1, x0
; D0 = a, D1 = 3.14
        fdiv   d0, d0, d1
; D0 = D0/D1 = a/3.14

        fmov   x1, d0
; X1 = a/3.14
        ldr    x2, [sp]
; X2 = b
        ldr    x0, .LC26
; X0 = 4.1
        fmov   d0, x2
; D0 = b
        fmov   d1, x0
; D1 = 4.1
        fmul   d0, d0, d1
; D0 = D0*D1 = b*4.1

        fmov   x0, d0

```

```

; X0 = D0 = b*4.1
    fmov    d0, x1
; D0 = a/3.14
    fmov    d1, x0
; D1 = X0 = b*4.1
    fadd    d0, d0, d1
; D0 = D0+D1 = a/3.14 + b*4.1

    fmov    x0, d0 ; \
    fmov    d0, x0 ; /
    add    sp, sp, 16
    ret

.LC25:
.word   1374389535      ; 3.14
.word   1074339512

.LC26:
.word   1717986918      ; 4.1
.word   1074816614

```

Sin optimización GCC . , GCC 4.9 . .

17.5.6 MIPS

[C.1.2 on page 1252.](#)

Listing 17.6: Con optimización GCC 4.4.5 (IDA)

```

f:
; $f12-$f13=A
; $f14-$f15=B
        lui     $v0, (dword_C4 >> 16) ; ?
; $f0:
        lwc1   $f0, dword_BC
        or     $at, $zero           ; load delay slot
        , NOP
; $f1:
        lwc1   $f1, $LC0
        lui     $v0, ($LC1 >> 16) ; ?
; A $f12-$f13, $f0-$f1, :
        div.d   $f0, $f12, $f0
; $f0-$f1=A/3.14
; $f2:
        lwc1   $f2, dword_C4
        or     $at, $zero           ; load delay slot
        , NOP
; $f3:

```

```

        lwc1    $f3, $LC1
        or      $at, $zero
; load delay slot2
        , NOP
; B   $f14-$f15, $f2-$f3, :
        mul.d   $f2, $f14, $f2
; $f2-$f3=B*4.1
        jr      $ra
; $f0-$f1:
        add.d   $f0, $f2
; branch delay ↴
        slot, NOP

.rodata.cst8:000000B8 $LC0:          .word 0x40091EB8
    ↴ # DATA XREF: f+C
.rodata.cst8:000000BC dword_BC:     .word 0x51EB851F
    ↴ # DATA XREF: f+4
.rodata.cst8:000000C0 $LC1:          .word 0x40106666
    ↴ # DATA XREF: f+10
.rodata.cst8:000000C4 dword_C4:     .word 0x66666666
    ↴ # DATA XREF: f

```

:

- LWC1
- DIV.D, MUL.D, ADD.D («.D» , «.S»)

²

17.6

```

#include <math.h>
#include <stdio.h>

int main ()
{
    printf ("32.01 ^ 1.54 = %lf\n", pow (32.01,1.54));

    return 0;
}

```

17.6.1 x86

(MSVC 2010):

²dennis(a)yurichev.com

Listing 17.7: MSVC 2010

```

CONST      SEGMENT
__real@40400147ae147ae1 DQ 040400147ae147ae1r      ; 32.01
__real@3ff8a3d70a3d70a4 DQ 03ff8a3d70a3d70a4r      ; 1.54
CONST      ENDS

_main     PROC
    push    ebp
    mov     ebp, esp
    sub    esp, 8 ;
    fld    QWORD PTR __real@3ff8a3d70a3d70a4
    fstp   QWORD PTR [esp]
    sub    esp, 8 ;
    fld    QWORD PTR __real@40400147ae147ae1
    fstp   QWORD PTR [esp]
    call    _pow
    add    esp, 8 ;

;

;

    fstp   QWORD PTR [esp] ;
    push    OFFSET $SG2651
    call    _printf
    add    esp, 12
    xor    eax, eax
    pop    ebp
    ret    0
_main     ENDP

```

: 17.6.2.

17.6.2 ARM + Sin optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

_main
var_C          = -0xC

        PUSH      {R7,LR}
        MOV       R7,SP
        SUB       SP,SP,#4
        VLDR    D16,=32.01
        VMOV    R0,R1,D16
        VLDR    D16,=1.54

```

	VMOV	R2, R3, D16
	BLX	_pow
	VMOV	D16, R0, R1
	MOV	R0, 0xFC1 ; "32.01 ^ 1.54 = %lf\n"
↳ \n"	ADD	R0, PC
	VMOV	R1, R2, D16
	BLX	_printf
	MOVS	R1, 0
	STR	R0, [SP,#0xC+var_C]
	MOV	R0, R1
	ADD	SP, SP, #4
	POP	{R7,PC}
dbl_2F90	DCFD 32.01	; DATA XREF: _main+6
dbl_2F98	DCFD 1.54	; DATA XREF: _main+E

_pow R0 y R1, R2 y R3. R0 y R1. _pow D16, R1 y R2, printf()

17.6.3 ARM + Sin optimización Keil 6/2013 (Modo ARM)

_main	STMFD	SP!, {R4-R6,LR}
	LDR	R2, =0xA3D70A4 ; y
	LDR	R3, =0x3FF8A3D7
	LDR	R0, =0xAE147AE1 ; x
	LDR	R1, =0x40400147
	BL	pow
	MOV	R4, R0
	MOV	R2, R4
	MOV	R3, R1
	ADR	R0, a32_011_54Lf ; "32.01 ^ 1.54 = %lf\n"
↳ n"	BL	_2printf
	MOV	R0, #0
	LDMFD	SP!, {R4-R6,PC}
y	DCD	0xA3D70A4 ; DATA XREF: _main+4
dword_520	DCD	0x3FF8A3D7 ; DATA XREF: _main+8
; double x		
x	DCD	0xAE147AE1 ; DATA XREF: _main+C
dword_528	DCD	0x40400147 ; DATA XREF: _main+10
a32_011_54Lf	DCB	"32.01 ^ 1.54 = %lf",0xA,0 ; DATA XREF: _main+24

17.6.4 ARM64 + Con optimización GCC (Linaro) 4.9

Listing 17.8: Con optimización GCC (Linaro) 4.9

```

f:
    stp    x29, x30, [sp, -16]!
    add    x29, sp, 0
    ldr    d1, .LC1 ; 1.54 D1
    ldr    d0, .LC0 ; 32.01 D0
    bl     pow
;  pow() D0
    adrp   x0, .LC2
    add    x0, x0, :lo12:.LC2
    bl     printf
    mov    w0, 0
    ldp    x29, x30, [sp], 16
    ret
.LC0:
; 32.01
    .word  -1374389535
    .word  1077936455
.LC1:
; 1.54
    .word  171798692
    .word  1073259479
.LC2:
    .string "32.01 ^ 1.54 = %lf\n"

```

D0 y D1: pow(). D0 pow(), printf(), printf().

17.6.5 MIPS

Listing 17.9: Con optimización GCC 4.4.5 (IDA)

```

main:
var_10      = -0x10
var_4       = -4
; :
    lui     $gp, (dword_9C >> 16)
    addiu  $sp, -0x20
    la     $gp, (__gnu_local_gp & 0xFFFF)
    sw     $ra, 0x20+var_4($sp)
    sw     $gp, 0x20+var_10($sp)
    lui     $v0, (dword_A4 >> 16) ; ?

```

```

; 32.01:           lwc1    $f12, dword_9C
;
; :               lw       $t9, (pow & 0xFFFF)($gp)
; 32.01:           lwc1    $f13, $LC0
;                 lui     $v0, ($LC1 >> 16) ; ?
; 1.54:            lwc1    $f14, dword_A4
;                 or      $at, $zero ; load delay slot, NOP
; 1.54:            lwc1    $f15, $LC1
; pow():           jalr    $t9
;                 or      $at, $zero ; branch delay slot, NOP
;                 lw      $gp, 0x20+var_10($sp)
;
; :
; mfc1    $a3, $f0
; lw       $t9, (printf & 0xFFFF)($gp)
; mfc1    $a2, $f1
; printf():        lui     $a0, ($LC2 >> 16) # "32.01 ^ 1.54 = %\n"
;                 ↳ lf\n"
;                 jalr    $t9
;                 la      $a0, ($LC2 & 0xFFFF) # "32.01 ^ 1.54 = %\n"
;                 ↳ %lf\n"
;
; :
; 0:               lw      $ra, 0x20+var_4($sp)
; move    $v0, $zero
; jr      $ra
; addiu   $sp, 0x20

.rodata.str1.4:00000084 $LC2:          .ascii "32.01 ^ 1.54 = %\n"
; ↳ %lf\n"<0>

; 32.01:
.rodata.cst8:00000098 $LC0:          .word 0x40400147
; ↳ # DATA XREF: main+20
.rodata.cst8:0000009C dword_9C:       .word 0xAE147AE1
; ↳ # DATA XREF: main
.rodata.cst8:0000009C
; ↳ # main+18
; 1.54:
.rodata.cst8:000000A0 $LC1:          .word 0x3FF8A3D7
; ↳ # DATA XREF: main+24

```

```
.rodata.cst8:000000A0
    ↳ # main+30
.rodata.cst8:000000A4 dword_A4:           .word 0xA3D70A4
    ↳ # DATA XREF: main+14
```

MFC1 («Move From Coprocessor 1»).

17.7

```
#include <stdio.h>

double d_max (double a, double b)
{
    if (a>b)
        return a;

    return b;
};

int main()
{
    printf ("%f\n", d_max (1.2, 3.4));
    printf ("%f\n", d_max (5.6, -4));
}
```

17.7.1 x86

Sin optimización MSVC

:

Listing 17.10: Sin optimización MSVC 2010

```
PUBLIC      _d_max
_TEXT      SEGMENT
_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_d_max      PROC
    push    ebp
    mov     ebp, esp
    fld     QWORD PTR _b$[ebp]
;
; : ST(0) = _b
```

```

; fcomp QWORD PTR _a$[ebp]
;

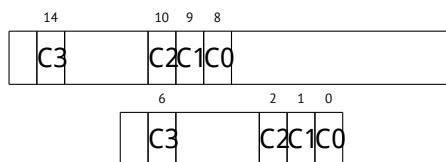
fNSTSW ax
test ah, 5
jp SHORT $LN1@d_max

; a>b

fld QWORD PTR _a$[ebp]
jmp SHORT $LN2@d_max
$LN1@d_max:
fld QWORD PTR _b$[ebp]
$LN2@d_max:
pop ebp
ret 0
_d_max ENDP

```

- 0, 0, 0.
- 0, 0, 1.
- 1, 0, 0.
- 1, 1, 1.



Wikipedia³:

One common reason to test the parity flag actually has nothing to do with parity. The FPU has four condition flags (C0 to C3), but they can not be tested directly, and must instead be first copied to the flags register. When this happens, C0 is placed in the carry flag, C2 in the parity flag and C3 in the zero flag. The C2 flag is set when e.g. incomparable floating point values (NaN or unsupported format) are compared with the FUCOM instructions.

³[wikipedia.org/wiki/Parity_flag](https://en.wikipedia.org/wiki/Parity_flag)

?

OllyDbg:

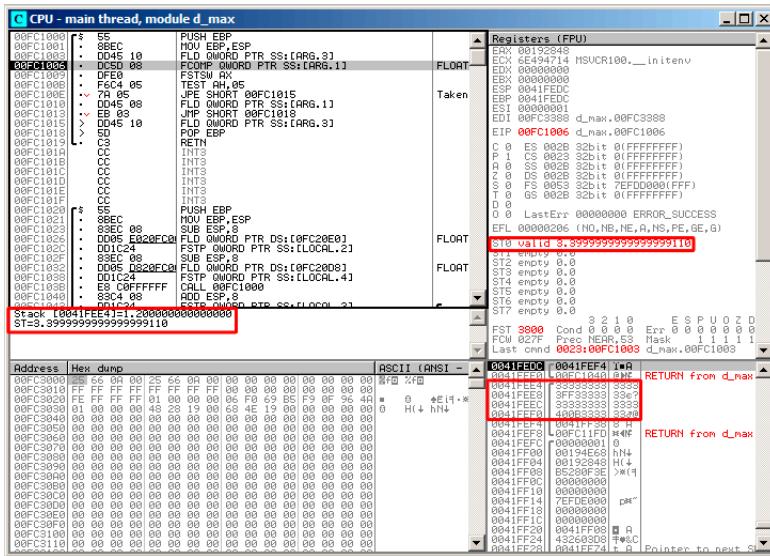


Figura 17.6: OllyDbg:

(.). ST(0).. OllyDbg.

FCOMP:

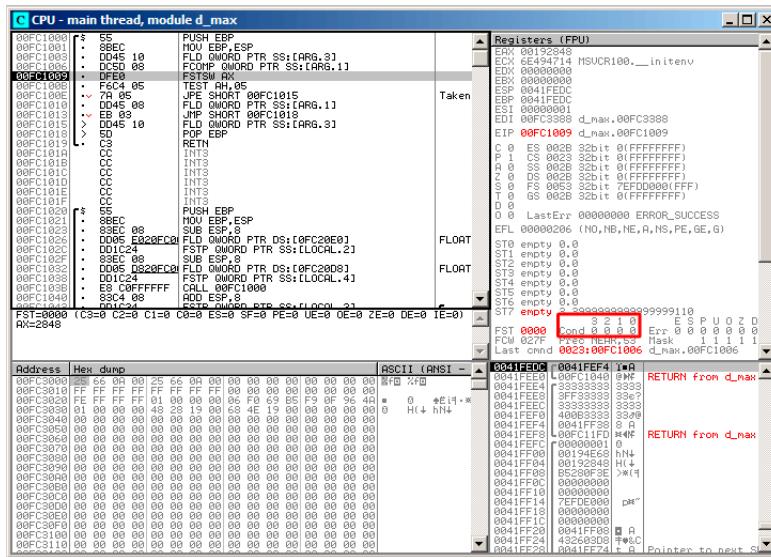


Figura 17.7: OllyDbg: FCOMP

[17.5.1 on page 277](#)

FNSTSW:

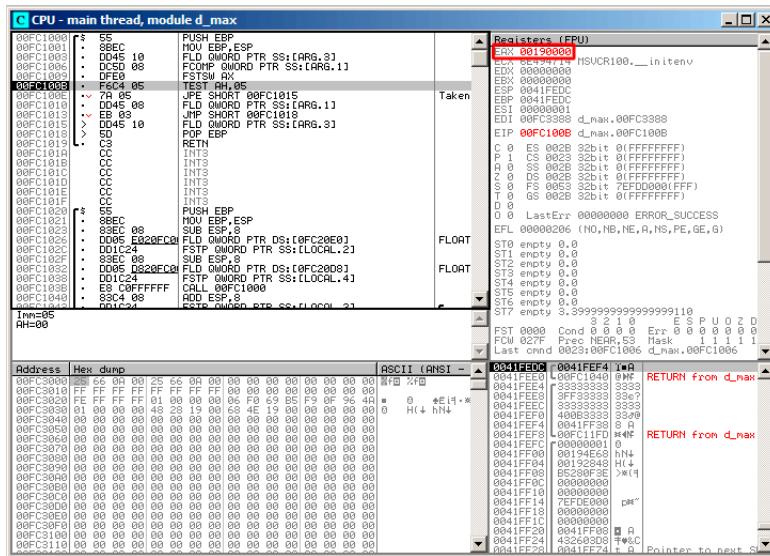


Figura 17.8: OllyDbg: FNSTSW

(OllyDbg FNSTSW FSTSW-).

TEST:

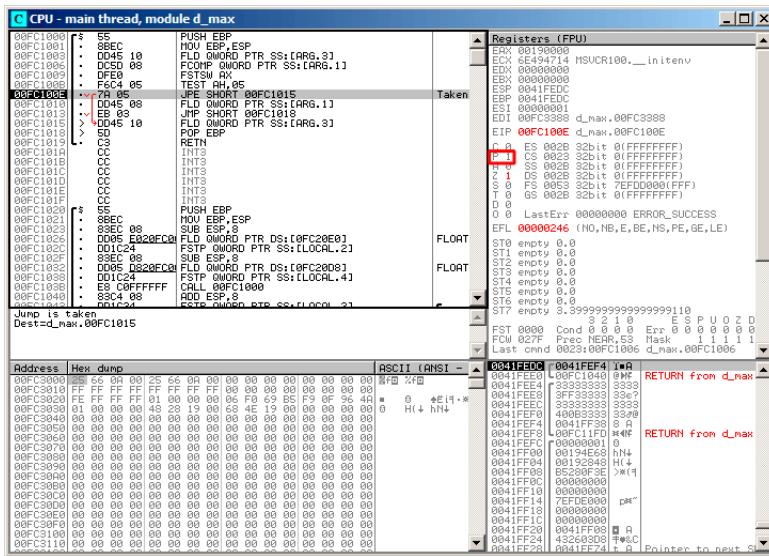


Figura 17.9: OllyDbg: TEST

OllyDbg JP JPE⁴ -..

⁴Jump Parity Even ()

JPE , FLD:

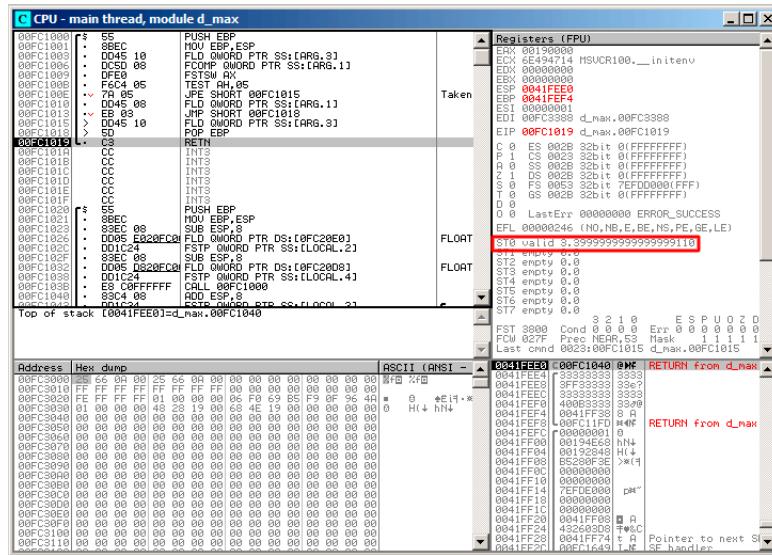


Figura 17.10: OllyDbg:

OllyDbg:

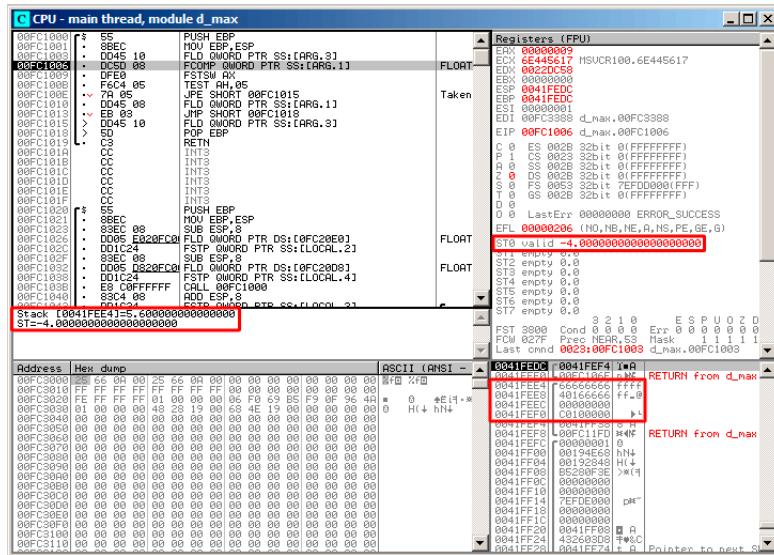


Figura 17.11: OllyDbg:

. b (-4) ST(0).. OllyDbg

FCOMP:

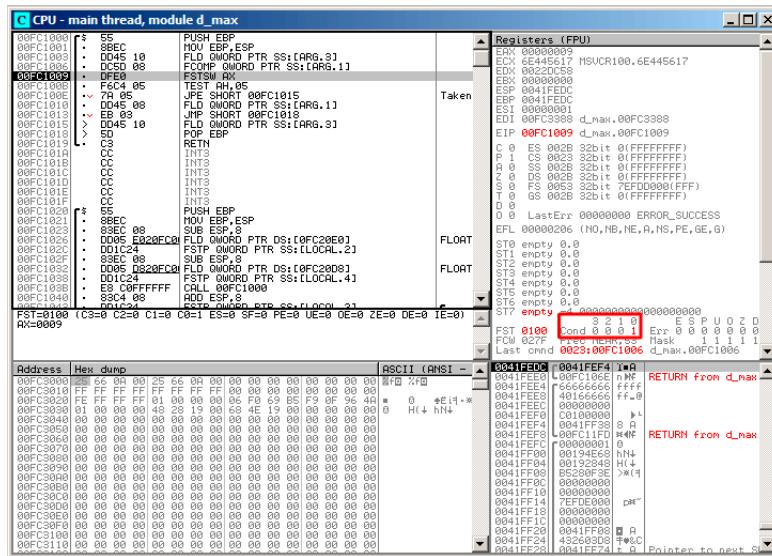


Figura 17.12: OllyDbg: FCOMP

FNSTSW:

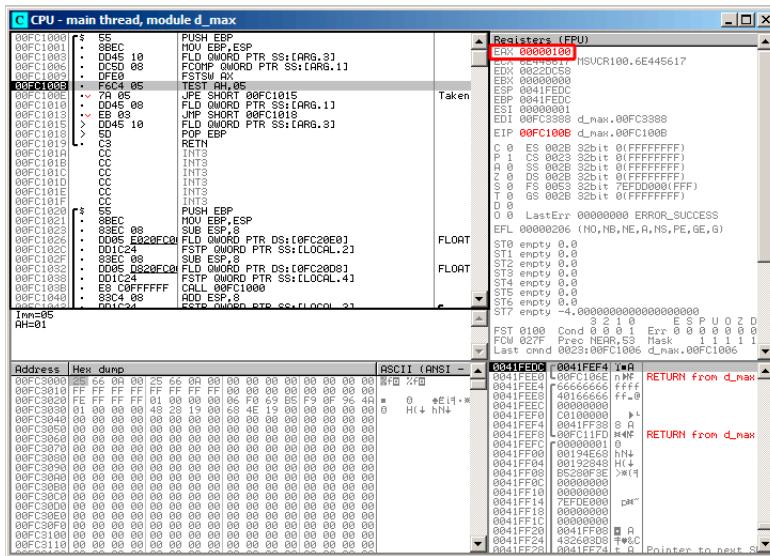


Figura 17.13: OllyDbg: FNSTSW

TEST:

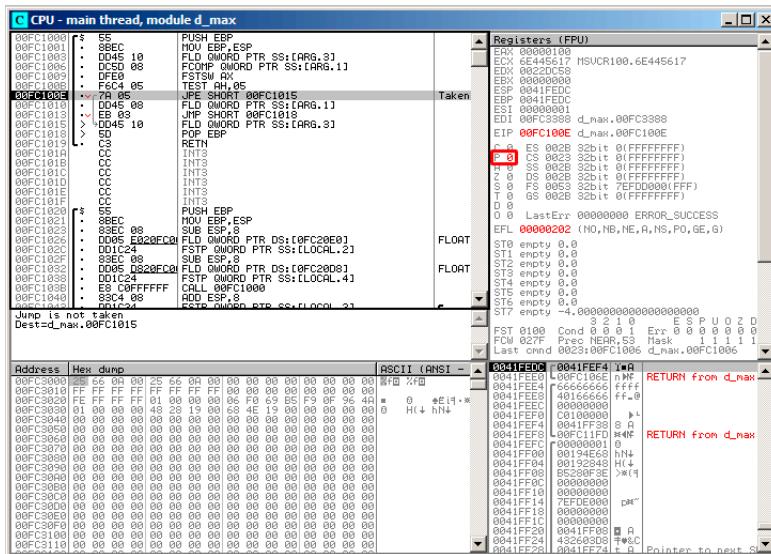


Figura 17.14: OllyDbg: TEST

PF.: JPE.

JPE FLD:

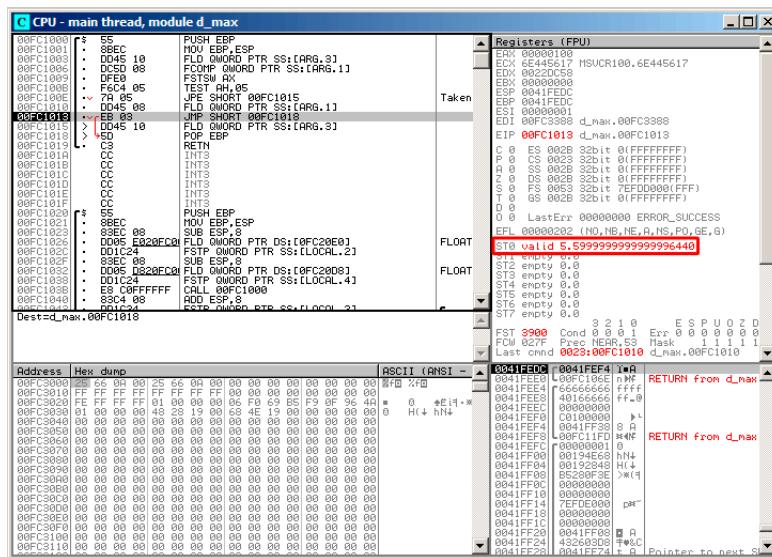


Figura 17.15: OllyDbg:

Con optimización MSVC 2010

Listing 17.11: Con optimización MSVC 2010

```

_a$ = 8           ; size = 8
_b$ = 16          ; size = 8
_d_max    PROC
    fld      QWORD PTR _b$[esp-4]
    fld      QWORD PTR _a$[esp-4]

; : ST(0) = _a, ST(1) = _b

    fcom    ST(1) ; _a ST(1) = (_b)
    fnstsw  ax
    test    ah, 65 ; 00000041H
    jne     SHORT $LN5@d_max
;

    fstp    ST(1)

```

```
; : ST(0) = _a
    ret    0
$LN5@d_max:
;
;
    fstp    ST(0)
; : ST(0) = _b
    ret    0
_d_max    ENDP
```

- 0, 0, 0.
- 0, 0, 1.
- 1, 0, 0.

FLD:

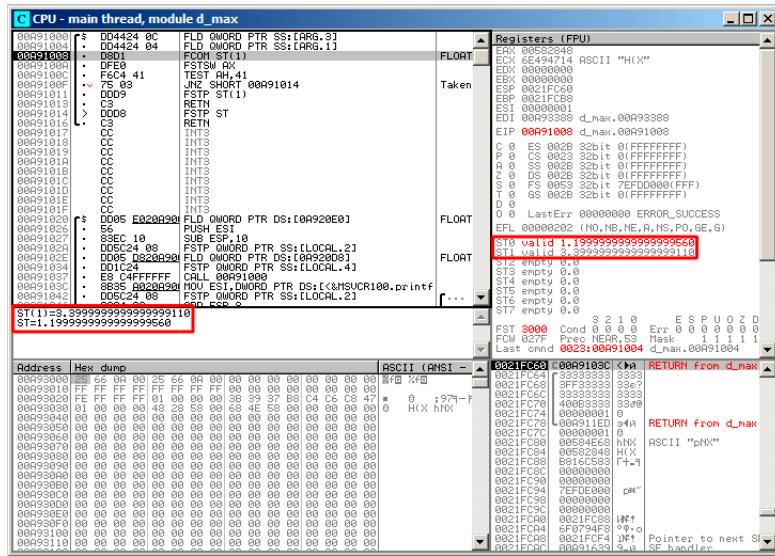


Figura 17.16: OllyDbg:

FCOM: OllyDbg ST(0) y ST(1) .

FCOM:

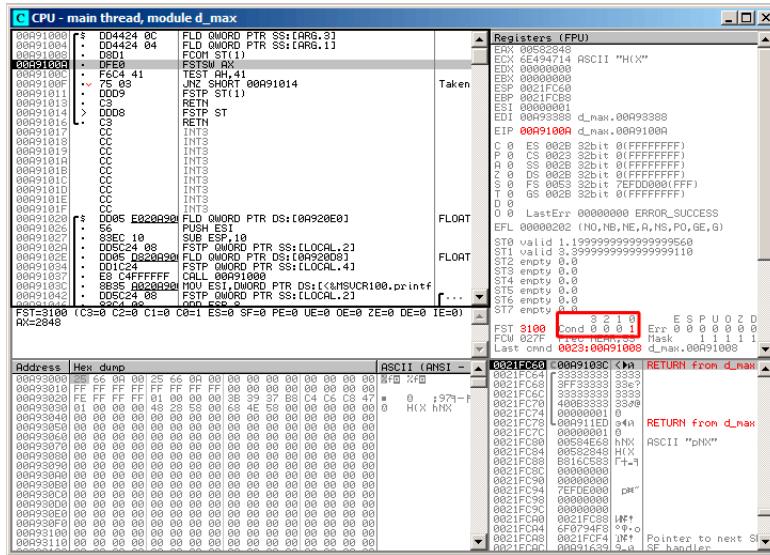


Figura 17.17: OllyDbg: FCOM

C0.

FNSTSW, AX=0x3100:

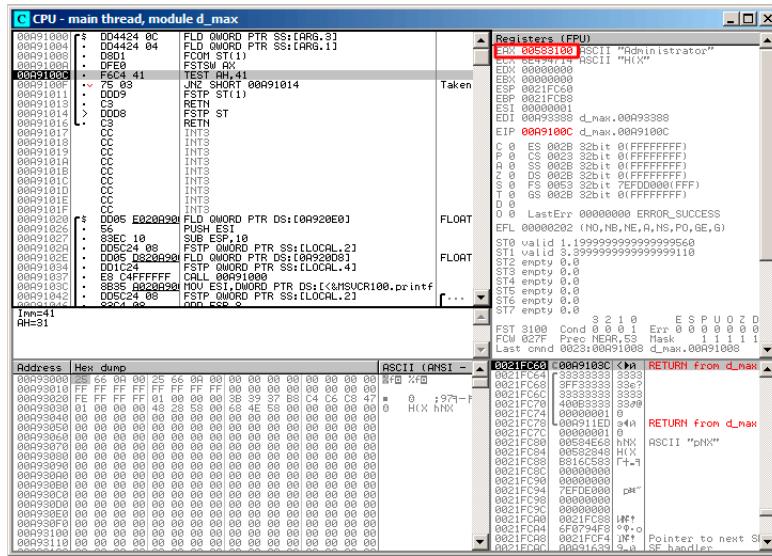


Figura 17.18: OllyDbg: FNSTSW

TEST:

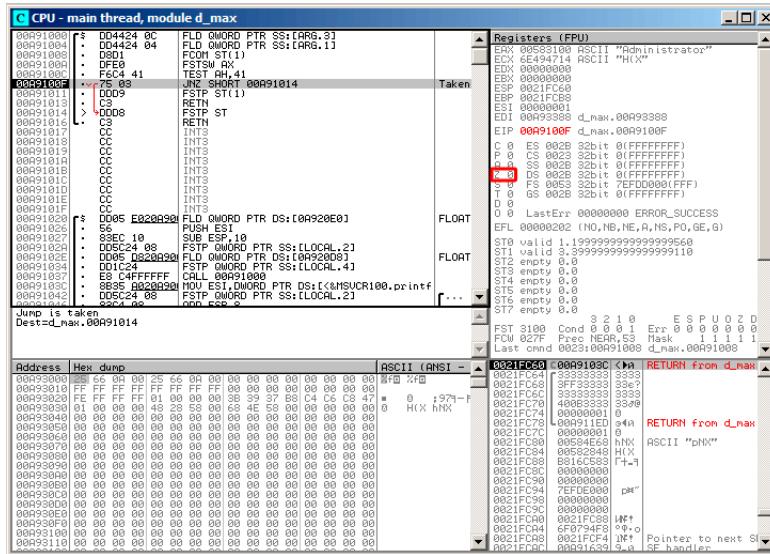


Figura 17.19: OllyDbg: TEST

ZF=0, .

FSTP ST (o FSTP ST(0)) :-

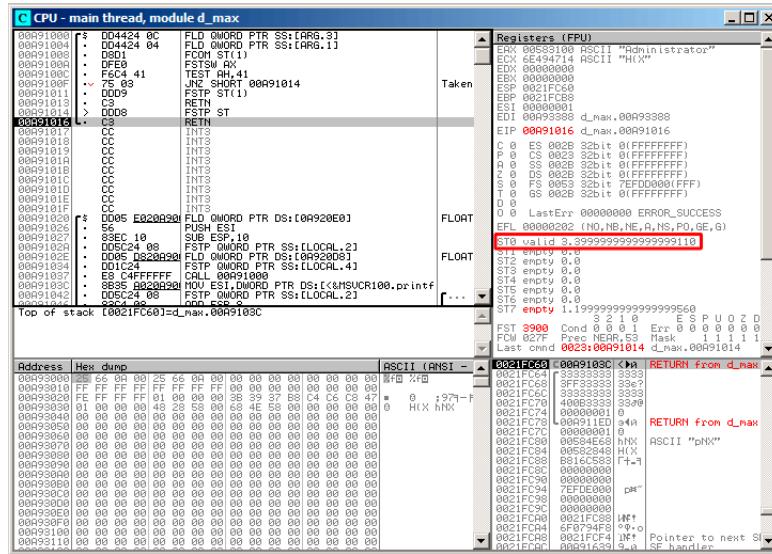


Figura 17.20: OllyDbg: FSTP

FSTP ST

FLD:

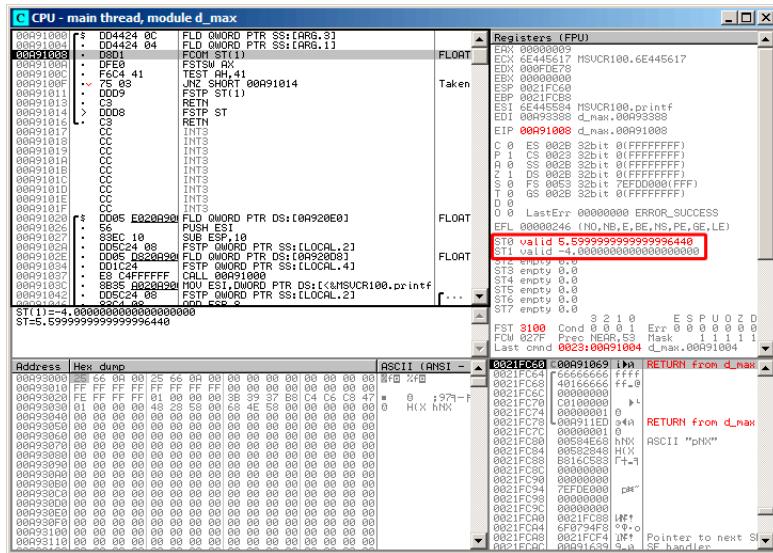


Figura 17.21: OllyDbg:

FCOM:

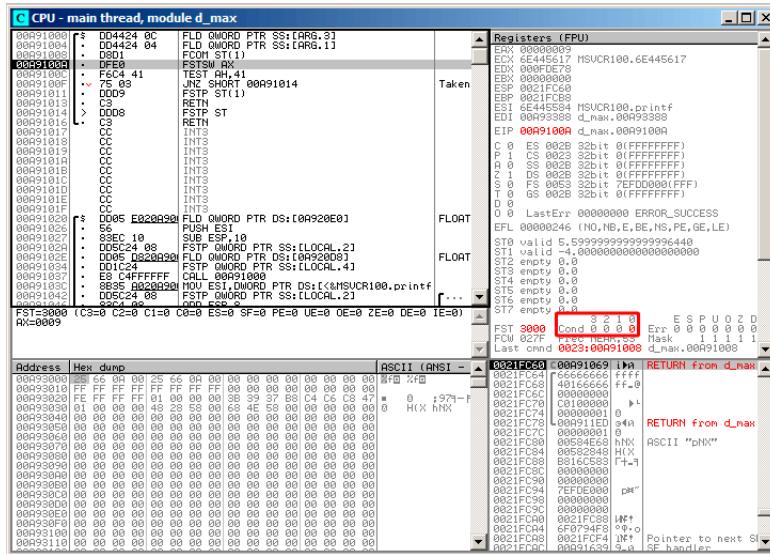


Figura 17.22: OllyDbg: FCOM

FNSTSW, AX=0x3000:

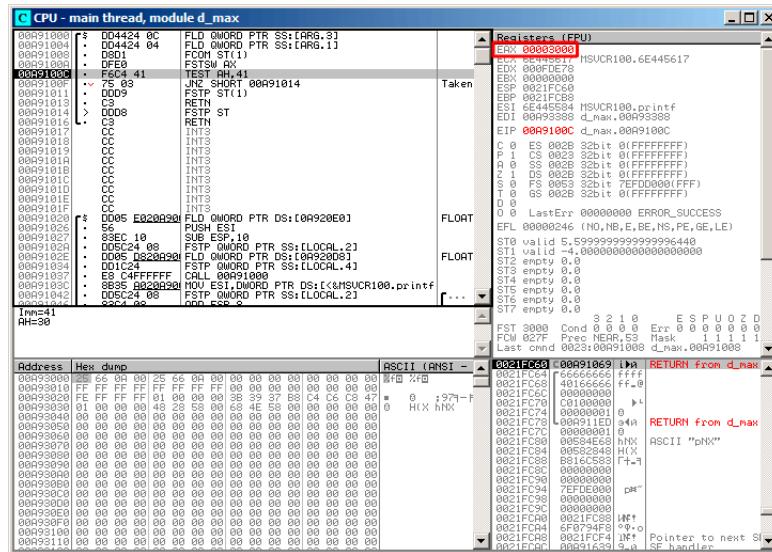


Figura 17.23: OllyDbg: FNSTSW

TEST:

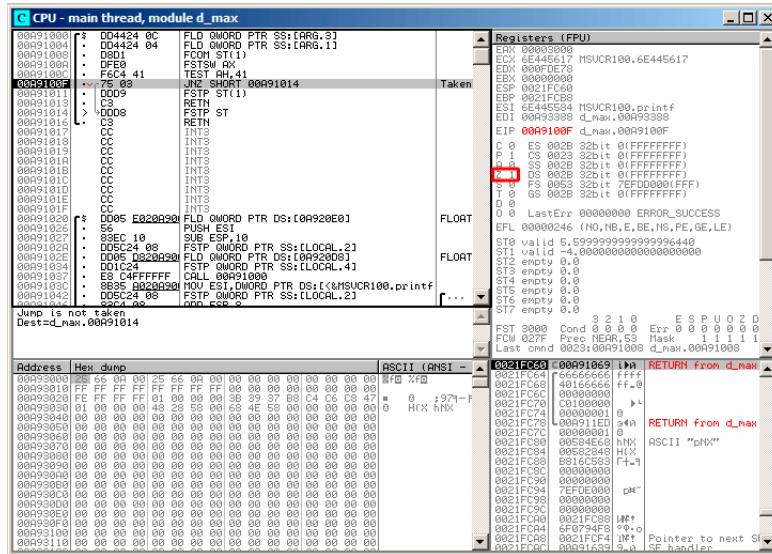


Figura 17.24: OllyDbg: TEST

ZF=1, .

FSTP ST(1) .

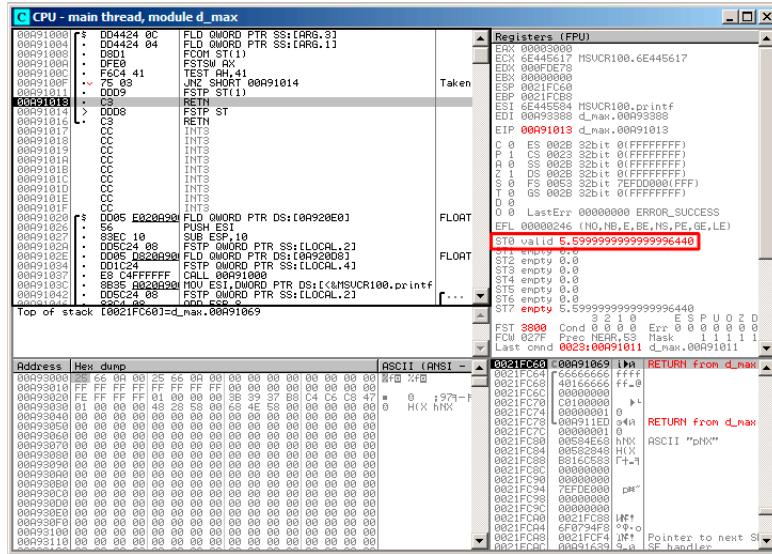


Figura 17.25: OllyDbg: FSTP

FSTP ST(1)

GCC 4.4.1

Listing 17.12: GCC 4.4.1

```
b          = qword ptr -10h
a          = qword ptr -8
a_first_half = dword ptr 8
a_second_half = dword ptr 0Ch
b_first_half = dword ptr 10h
b_second_half = dword ptr 14h

push    ebp
mov     ebp, esp
sub    esp, 10h

; :

mov    eax, [ebp+ta first half]
```

```

    mov      dword ptr [ebp+a], eax
    mov      eax, [ebp+a_second_half]
    mov      dword ptr [ebp+a+4], eax
    mov      eax, [ebp+b_first_half]
    mov      dword ptr [ebp+b], eax
    mov      eax, [ebp+b_second_half]
    mov      dword ptr [ebp+b+4], eax

; :

fld      [ebp+a]
fld      [ebp+b]

; : ST(0) - b; ST(1) - a

fxch    st(1) ;

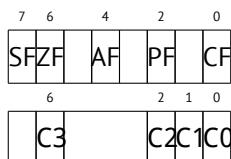
; : ST(0) - a; ST(1) - b

fucompp   ;
fnstsw  ax ;
sahf     ;
setnbe  al ;
test    al, al           ; AL==0 ?
jz      short loc_8048453 ;
fld      [ebp+a]
jmp     short locret_8048456

loc_8048453:
fld      [ebp+b]

locret_8048456:
leave
retn
d_max endp

```



- 0, 0, 0.
- 0, 0, 1.
- 1, 0, 0.

- ZF=0, PF=0, CF=0.
- ZF=0, PF=0, CF=1.
- ZF=1, PF=0, CF=0.

Con optimización GCC 4.4.1

Listing 17.13: Con optimización GCC 4.4.1

```

d_max          public d_max
                proc near
arg_0          = qword ptr  8
arg_8          = qword ptr  10h

                push    ebp
                mov     ebp, esp
                fld     [ebp+arg_0] ; _a
                fld     [ebp+arg_8] ; _b

; : ST(0) = _b, ST(1) = _a
                fxch   st(1)

; : ST(0) = _a, ST(1) = _b
                fucom  st(1) ;
                fnstsw ax
                sahf
                ja     short loc_8048448

;
;

                fstp   st
                jmp    short loc_804844A

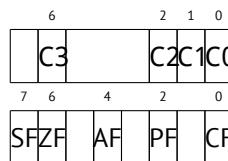
loc_8048448:
;
                fstp   st(1)

loc_804844A:
                pop    ebp
                retn
d_max          endp

```

Es casi lo mismo excepto que se usa JA después de SAHF. De hecho, las instrucciones de salto condicional que verifican «mayor», «menor» o «igual» para comparación de números sin signo (éstas son JA, JAE, JB, JBE, JEJZ, JNA, JNAE, JNB,

JNBE, JNEJNZ) verifican solo las banderas CF y ZF.



GCC 4.8.1

⁵. FUCOMI () y FCMOVcc (CMOVcc,). .

:

Listing 17.14: Con optimización GCC 4.8.1

```

fld      QWORD PTR [esp+4]      ;  "a"
fld      QWORD PTR [esp+12]      ;  "b"
; ST0=b, ST1=a
fxch    st(1)
; ST0=a, ST1=b
; "a"  "b"
fucomi  st, st(1)
; ST1 ()  ST0  a<=b
;
fcmovbe st, st(1)
; ST1
fstp    st(1)
ret

```

FUCOMI ST(0) (a) y ST(1) (b). FCMOVBE ST(1) () ST(0) () ST0(a) <= ST1(b).
 $(a > b)$, a en ST(0).

FSTP ST(0) ST(1).

GDB:

Listing 17.15: Con optimización GCC 4.8.1 and GDB

```

1 dennis@ubuntuvm:~/polygon$ gcc -O3 d_max.c -o d_max -fno-inline
2 dennis@ubuntuvm:~/polygon$ gdb d_max
3 GNU gdb (GDB) 7.6.1-ubuntu
4 Copyright (C) 2013 Free Software Foundation, Inc.

```

⁵Pentium Pro, Pentium-II, Spanish text placeholder.

```

5 License GPLv3+: GNU GPL version 3 or later <http://gnu.org/遵守>
   ↳ licenses/gpl.html>
6 This is free software: you are free to change and redistribute ↳
   ↳ it.
7 There is NO WARRANTY, to the extent permitted by law. Type "↙
   ↳ show copying"
8 and "show warranty" for details.
9 This GDB was configured as "i686-linux-gnu".
10 For bug reporting instructions, please see:
11 <http://www.gnu.org/software/gdb/bugs/>...
12 Reading symbols from /home/dennis/polygon/d_max...(no debugging) ↳
   ↳ symbols found)...done.
13 (gdb) b d_max
14 Breakpoint 1 at 0x80484a0
15 (gdb) run
16 Starting program: /home/dennis/polygon/d_max
17
18 Breakpoint 1, 0x080484a0 in d_max ()
19 (gdb) ni
20 0x080484a4 in d_max ()
21 (gdb) disas $eip
22 Dump of assembler code for function d_max:
23 0x080484a0 <+0>:    fldl   0x4(%esp)
24 => 0x080484a4 <+4>:    fldl   0xc(%esp)
25 0x080484a8 <+8>:    fxch   %st(1)
26 0x080484aa <+10>:   fucomi %st(1),%st
27 0x080484ac <+12>:   fcmovbe %st(1),%st
28 0x080484ae <+14>:   fstp    %st(1)
29 0x080484b0 <+16>:   ret
30 End of assembler dump.
31 (gdb) ni
32 0x080484a8 in d_max ()
33 (gdb) info float
34   R7: Valid    0x3fff999999999999800 +1.19999999999999956
35 =>R6: Valid    0x4000d99999999999800 +3.39999999999999911
36   R5: Empty    0x00000000000000000000000000000000
37   R4: Empty    0x00000000000000000000000000000000
38   R3: Empty    0x00000000000000000000000000000000
39   R2: Empty    0x00000000000000000000000000000000
40   R1: Empty    0x00000000000000000000000000000000
41   R0: Empty    0x00000000000000000000000000000000
42
43 Status Word:      0x3000
44                      TOP: 6
45 Control Word:     0x037f  IM DM ZM OM UM PM
46                      PC: Extended Precision (64-bits)
47                      RC: Round to nearest

```

```

48 Tag Word:          0xffff
49 Instruction Pointer: 0x73:0x080484a4
50 Operand Pointer:   0xb:0xbfffff118
51 Opcode:           0x0000
52 (gdb) ni
53 0x080484aa in d_max ()
54 (gdb) info float
55     R7: Valid    0x4000d9999999999999800 +3.399999999999999911
56 =>R6: Valid    0x3fff99999999999999800 +1.19999999999999956
57     R5: Empty    0x00000000000000000000000000000000
58     R4: Empty    0x00000000000000000000000000000000
59     R3: Empty    0x00000000000000000000000000000000
60     R2: Empty    0x00000000000000000000000000000000
61     R1: Empty    0x00000000000000000000000000000000
62     R0: Empty    0x00000000000000000000000000000000
63
64 Status Word:      0x3000
65                 TOP: 6
66 Control Word:     0x037f IM DM ZM OM UM PM
67                 PC: Extended Precision (64-bits)
68                 RC: Round to nearest
69 Tag Word:          0xffff
70 Instruction Pointer: 0x73:0x080484a8
71 Operand Pointer:   0xb:0xbfffff118
72 Opcode:           0x0000
73 (gdb) disas $eip
74 Dump of assembler code for function d_max:
75 0x080484a0 <+0>:   fldl  0x4(%esp)
76 0x080484a4 <+4>:   fldl  0xc(%esp)
77 0x080484a8 <+8>:   fxch  %st(1)
78 => 0x080484aa <+10>: fucomi %st(1),%st
79 0x080484ac <+12>: fcmovbe %st(1),%st
80 0x080484ae <+14>: fstp   %st(1)
81 0x080484b0 <+16>: ret
82 End of assembler dump.
83 (gdb) ni
84 0x080484ac in d_max ()
85 (gdb) info registers
86 eax          0x1      1
87 ecx          0xbfffff1c4 -1073745468
88 edx          0x8048340 134513472
89 ebx          0xb7fbf000 -1208225792
90 esp          0xbfffff10c 0xbfffff10c
91 ebp          0xbfffff128 0xbfffff128
92 esi          0x0      0
93 edi          0x0      0
94 eip          0x80484ac <d_max+12>

```

```

95 eflags          0x203      [ CF IF ]
96 cs              0x73       115
97 ss              0x7b       123
98 ds              0x7b       123
99 es              0x7b       123
100 fs             0x0        0
101 gs              0x33       51
102 (gdb) ni
103 0x080484ae in d_max ()
104 (gdb) info float
105   R7: Valid    0x4000d99999999999999800 +3.399999999999999911
106 =>R6: Valid    0x4000d99999999999999800 +3.399999999999999911
107   R5: Empty    0x00000000000000000000000000000000
108   R4: Empty    0x00000000000000000000000000000000
109   R3: Empty    0x00000000000000000000000000000000
110   R2: Empty    0x00000000000000000000000000000000
111   R1: Empty    0x00000000000000000000000000000000
112   R0: Empty    0x00000000000000000000000000000000
113
114 Status Word:      0x3000
115                  TOP: 6
116 Control Word:     0x037f   IM DM ZM OM UM PM
117                  PC: Extended Precision (64-bits)
118                  RC: Round to nearest
119 Tag Word:          0xffff
120 Instruction Pointer: 0x73:0x080484ac
121 Operand Pointer:   0x7b:0xbfffff118
122 Opcode:           0x0000
123 (gdb) disas $eip
124 Dump of assembler code for function d_max:
125 0x080484a0 <+0>:   fldl   0x4(%esp)
126 0x080484a4 <+4>:   fldl   0xc(%esp)
127 0x080484a8 <+8>:   fxch   %st(1)
128 0x080484aa <+10>:  fucomi %st(1),%st
129 0x080484ac <+12>:  fcmovbe %st(1),%st
130 => 0x080484ae <+14>: fstp   %st(1)
131 0x080484b0 <+16>:  ret
132 End of assembler dump.
133 (gdb) ni
134 0x080484b0 in d_max ()
135 (gdb) info float
136 =>R7: Valid    0x4000d99999999999999800 +3.399999999999999911
137   R6: Empty    0x4000d99999999999999800
138   R5: Empty    0x00000000000000000000000000000000
139   R4: Empty    0x00000000000000000000000000000000
140   R3: Empty    0x00000000000000000000000000000000
141   R2: Empty    0x00000000000000000000000000000000

```

```

142 R1: Empty      0x00000000000000000000000000000000
143 R0: Empty      0x00000000000000000000000000000000
144
145 Status Word:    0x3800
146                  TOP: 7
147 Control Word:   0x037f IM DM ZM OM UM PM
148                  PC: Extended Precision (64-bits)
149                  RC: Round to nearest
150 Tag Word:        0x3fff
151 Instruction Pointer: 0x73:0x080484ae
152 Operand Pointer: 0x7b:0xbfffff118
153 Opcode:          0x0000
154 (gdb) quit
155 A debugging session is active.
156
157 Inferior 1 [process 30194] will be killed.
158
159 Quit anyway? (y or n) y
160 dennis@ubuntuvm:~/polygon$
```

«ni»,

(línea 33).

([17.5.1 on page 277](#)). GDB STx (Rx). (35)

(línea 54).

FUCOMI (línea 83). :CF (línea 95).

FCMOVBE

FSTP (línea 136)..

17.7.2 ARM

Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Listing 17.16: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

VMOV	D16, R2, R3 ; b
VMOV	D17, R0, R1 ; a
VCMPE.F64	D17, D16
VMRS	APSR_nzcv, FPSCR
VMOVGT.F64	D16, D17 ; "b" D16
VMOV	R0, R1, D16
BX	LR

D17 y D16 VCMPE. ([FPSCR⁶](#)), .,. VMRS (N, Z, C, V)

VMOVGT MOVTG, (*GTGreater Than*).

, D17.

Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

Listing 17.17: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

VMOV          D16, R2, R3 ; b
VMOV          D17, R0, R1 ; a
VCMPE.F64    D17, D16
VMRS          APSR_nzcv, FPSCR
IT GT
VMOVGT.F64   D16, D17
VMOV          R0, R1, D16
BX

```

,

IT *if-then block*. IT GT GT (*Greater Than*).

Angry Birds (iOS):

Listing 17.18: Angry Birds Classic

```

...
ITE NE
VMOVNE        R2, R3, D16
VMOVEQ        R2, R3, D17
BLX           __objc_msgSend ; not prefixed
...

```

ITE *if-then-else* ITE (*NE, not equal*), .(*NE EQ (equal)*).

, Angry Birds:

Listing 17.19: Angry Birds Classic

```

...
ITTTT EQ
MOVEQ         R0, R4
ADDEQ         SP, SP, #0x20
POPEQ.W       {R8,R10}
POPEQ         {R4-R7,PC}
BLX           __stack_chk_fail ; not prefixed
...

```

⁶(ARM) Floating-Point Status and Control Register

-EQ.

ITEEE EQ (*if-then-else-else-else*),

```
-EQ
-NE
-NE
-NE
```

Angry Birds:

Listing 17.20: Angry Birds Classic

```
...
CMP.W      R0, #0xFFFFFFFF
ITTE LE
SUBLE.W    R10, R0, #1
NEGLE      R0, R0
MOVG T    R10, R0
MOVS       R6, #0          ; not prefixed
CBZ        R0, loc_1E7E32 ; not prefixed
...
```

ITTE (*if-then-then-else*)

: IT, ITE, ITT, ITTE, ITTT, ITTTT.

```
cat AngryBirdsClassic.lst | grep " BF" | grep "IT" > results.✓
↳ lst
```

Sin optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Listing 17.21: Sin optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```
b           = -0x20
a           = -0x18
val_to_return = -0x10
saved_R7     = -4

STR         R7, [SP,#saved_R7]!
MOV         R7, SP
SUB         SP, SP, #0x1C
BIC         SP, SP, #7
VMOV        D16, R2, R3
VMOV        D17, R0, R1
VSTR        D17, [SP,#0x20+a]
VSTR        D16, [SP,#0x20+b]
VLDR        D16, [SP,#0x20+a]
```

	VLDR	D17, [SP,#0x20+b]
	VCMPE.F64	D16, D17
	VMRS	APSR_nzcv, FPSCR
	BLE	loc_2E08
	VLDR	D16, [SP,#0x20+a]
	VSTR	D16, [SP,#0x20+val_to_return]
	B	loc_2E10
loc_2E08		
	VLDR	D16, [SP,#0x20+b]
	VSTR	D16, [SP,#0x20+val_to_return]
loc_2E10		
	VLDR	D16, [SP,#0x20+val_to_return]
	VMOV	R0, R1, D16
	MOV	SP, R7
	LDR	R7, [SP+0x20+b],#4
	BX	LR

,

Con optimización Keil 6/2013 (Modo Thumb)

Listing 17.22: Con optimización Keil 6/2013 (Modo Thumb)

	PUSH	{R3-R7,LR}
	MOVS	R4, R2
	MOVS	R5, R3
	MOVS	R6, R0
	MOVS	R7, R1
	BL	__aeabi_cdrcmple
	BCS	loc_1C0
	MOVS	R0, R6
	MOVS	R1, R7
	POP	{R3-R7,PC}
loc_1C0		
	MOVS	R0, R4
	MOVS	R1, R5
	POP	{R3-R7,PC}

__aeabi_cdrcmple.

N.B. BCS (*Carry set—Greater than or equal*)

17.7.3 ARM64

Con optimización GCC (Linaro) 4.9

```
d_max:
; D0 - a, D1 - b
    fcmpe    d0, d1
    fcsel    d0, d0, d1, gt
; D0
    ret
```

ARM64 ISA [APSR](#)⁷ [FPSCR](#) . FPU⁸ . FCMPE. D0 y D1 () [APSR](#) (N, Z, C, V).

FCSEL (*Floating Conditional Select*) D0 o D1 D0 (GTGreater Than), [APSR](#) [FPSCR](#).
(GT), D0 D0 (). D1 D0.

Sin optimización GCC (Linaro) 4.9

```
d_max:
; "Register Save Area"
    sub      sp, sp, #16
    str     d0, [sp,8]
    str     d1, [sp]
;
    ldr      x1, [sp,8]
    ldr      x0, [sp]
    fmov    d0, x1
    fmov    d1, x0
; D0 - a, D1 - b
    fcmpe   d0, d1
    ble     .L76
; a>b; D0 (a) X0
    ldr      x0, [sp,8]
    b       .L74
.L76:
; a<=b; D1 (b) X0
    ldr      x0, [sp]
.L74:
; X0
    fmov    d0, x0
; D0
    add     sp, sp, 16
```

⁷(ARM) Application Program Status Register

⁸Floating-point unit

```
ret
```

. (*Register Save Area*). X0/X1 D0/D1 FCMPE.. FCMPE [APSR](#). FCSEL: BLE (*Branch if Less than or Equal*). (a > b) a X0. (a <= b) b X0. X0 D0, .

Ejercicio

Con optimización GCC (Linaro) 4.9float

```
float f_max (float a, float b)
{
    if (a>b)
        return a;

    return b;
};
```

```
f_max:
; S0 - a, S1 - b
    fcmpe    s0, s1
    fcsel    s0, s0, s1, gt
; S0
    ret
```

17.7.4 MIPS

Listing 17.23: Con optimización GCC 4.4.5 (IDA)

```
d_max:
; $f14<$f12 (b<a):
    c.lt.d  $f14, $f12
    or      $at, $zero ; NOP
; locret_14
    bc1t    locret_14
; ("a"):
    mov.d   $f0, $f12 ; branch delay slot
;
; "b":
    mov.d   $f0, $f14
```

locret_14:

jr	\$ra
or	\$at, \$zero ; branch delay slot, NOP

C.LT.D. LT «Less Than». D *double*.

BC1T. T («True»). «BC1F» («False»).

17.8

17.9 x64

17.10 Ejercicios

17.10.1 Ejercicio #1

FXCH [17.7.1 on page 313](#).

17.10.2 Ejercicio #2

Lo que hace el código?

Listing 17.24: Con optimización MSVC 2010

```

__real@4014000000000000 DQ 0401400000000000r ; 5

_a1$ = 8          ; size = 8
_a2$ = 16         ; size = 8
_a3$ = 24         ; size = 8
_a4$ = 32         ; size = 8
_a5$ = 40         ; size = 8
_f      PROC
    fld    QWORD PTR _a1$[esp-4]
    fadd   QWORD PTR _a2$[esp-4]
    fadd   QWORD PTR _a3$[esp-4]
    fadd   QWORD PTR _a4$[esp-4]
    fadd   QWORD PTR _a5$[esp-4]
    fdiv   QWORD PTR __real@4014000000000000
    ret    0
_f      ENDP

```

Listing 17.25: Sin optimización Keil 6/2013 (Modo Thumb / Cortex-R4F CPU)

f PROC	VADD.F64 d0,d0,d1
--------	-------------------

```

VMOV.F64 d1,#5.00000000
VADD.F64 d0,d0,d2
VADD.F64 d0,d0,d3
VADD.F64 d2,d0,d4
VDIV.F64 d0,d2,d1
BX      lr
ENDP

```

Listing 17.26: Con optimización GCC 4.9 (ARM64)

```

f:
    fadd    d0, d0, d1
    fmov    d1, 5.0e+0
    fadd    d2, d0, d2
    fadd    d3, d2, d3
    fadd    d0, d3, d4
    fdiv    d0, d0, d1
    ret

```

Listing 17.27: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:
arg_10      = 0x10
arg_14      = 0x14
arg_18      = 0x18
arg_1C      = 0x1C
arg_20      = 0x20
arg_24      = 0x24

        lwc1    $f0, arg_14($sp)
        add.d   $f2, $f12, $f14
        lwc1    $f1, arg_10($sp)
        lui     $v0, ($LC0 >> 16)
        add.d   $f0, $f2, $f0
        lwc1    $f2, arg_1C($sp)
        or      $at, $zero
        lwc1    $f3, arg_18($sp)
        or      $at, $zero
        add.d   $f0, $f2
        lwc1    $f2, arg_24($sp)
        or      $at, $zero
        lwc1    $f3, arg_20($sp)
        or      $at, $zero
        add.d   $f0, $f2
        lwc1    $f2, dword_6C
        or      $at, $zero
        lwc1    $f3, $LC0

```

	jr	\$ra
	div.d	\$f0, \$f2
\$LC0:	.word	0x40140000 # DATA XREF: f+C # f+44
dword_6C:	.word	0 # DATA XREF: f+3C

Responda [G.1.11 on page 1262](#).

Capítulo 18

Matriz

18.1

```
#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<20; i++)
        a[i]=i*2;

    for (i=0; i<20; i++)
        printf ("a[%d]=%d\n", i, a[i]);

    return 0;
}
```

18.1.1 x86

MSVC

:

Listing 18.1: MSVC 2008

```
_TEXT      SEGMENT
_i$ = -84           ; size = 4
_a$ = -80          ; size = 80
```

```

_main      PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 84          ; 00000054H
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN6@main
$LN5@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN6@main:
    cmp     DWORD PTR _i$[ebp], 20      ; 00000014H
    jge     SHORT $LN4@main
    mov     ecx, DWORD PTR _i$[ebp]
    shl     ecx, 1
    mov     edx, DWORD PTR _i$[ebp]
    mov     DWORD PTR _a$[ebp+edx*4], ecx
    jmp     SHORT $LN5@main
$LN4@main:
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp     DWORD PTR _i$[ebp], 20      ; 00000014H
    jge     SHORT $LN1@main
    mov     ecx, DWORD PTR _i$[ebp]
    mov     edx, DWORD PTR _a$[ebp+ecx*4]
    push    edx
    mov     eax, DWORD PTR _i$[ebp]
    push    eax
    push    OFFSET $SG2463
    call    _printf
    add     esp, 12          ; 0000000cH
    jmp     SHORT $LN2@main
$LN1@main:
    xor     eax, eax
    mov     esp, ebp
    pop     ebp
    ret     0
_main      ENDP

```

OllyDbg.

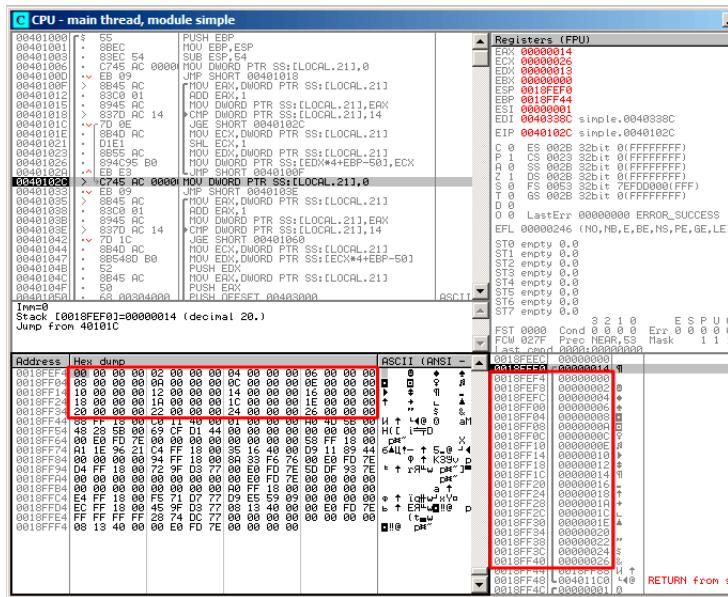


Figura 18.1: OllyDbg:

GCC

Listing 18.2: GCC 4.4.1

```

public main
main proc near ; DATA XREF: _start+17

var_70      = dword ptr -70h
var_6C      = dword ptr -6Ch
var_68      = dword ptr -68h
i_2         = dword ptr -54h
i           = dword ptr -4

push    ebp
mov     ebp, esp
and    esp, 0FFFFFFF0h
sub    esp, 70h
mov     [esp+70h+i], 0 ; i=0
jmp     short loc_804840A

```

```

loc_80483F7:
    mov     eax, [esp+70h+i]
    mov     edx, [esp+70h+i]
    add     edx, edx          ; edx=i*2
    mov     [esp+eax*4+70h+i_2], edx
    add     [esp+70h+i], 1      ; i++

loc_804840A:
    cmp     [esp+70h+i], 13h
    jle     short loc_80483F7
    mov     [esp+70h+i], 0
    jmp     short loc_8048441

loc_804841B:
    mov     eax, [esp+70h+i]
    mov     edx, [esp+eax*4+70h+i_2]
    mov     eax, offset aADD ; "a[%d]=%d\n"
    mov     [esp+70h+var_68], edx
    mov     edx, [esp+70h+i]
    mov     [esp+70h+var_6C], edx
    mov     [esp+70h+var_70], eax
    call    _printf
    add     [esp+70h+i], 1

loc_8048441:
    cmp     [esp+70h+i], 13h
    jle     short loc_804841B
    mov     eax, 0
    leave
    retn
main
    endp

```

18.1.2 ARM

Sin optimización Keil 6/2013 (Modo ARM)

```

EXPORT _main
_main
    STMFD   SP!, {R4,LR}
    SUB    SP, SP, #0x50      ;
;

    MOV     R4, #0             ; i
    B      loc_4A0
loc_494

```

```

        MOV      R0, R4, LSL#1      ; R0=R4*2
        STR      R0, [SP,R4,LSL#2]  ;
        ADD      R4, R4, #1       ; i=i+1

loc_4A0
        CMP      R4, #20          ; i<20?
        BLT      loc_494         ;
;

        MOV      R4, #0           ; i
        B       loc_4C4

loc_4B0
        LDR      R2, [SP,R4,LSL#2] ;
        MOV      R1, R4           ; () R1=i
        ADR      R0, aADD         ; "a[%d]=%d\n"
        BL       __2printf
        ADD      R4, R4, #1       ; i=i+1

loc_4C4
        CMP      R4, #20          ; i<20?
        BLT      loc_4B0         ;
        MOV      R0, #0           ;
        ADD      SP, SP, #0x50    ;
        LDMFD   SP!, {R4,PC}

```

SUB SP, SP, #0x50
STR R0, [SP,R4,LSL#2].

Con optimización Keil 6/2013 (Modo Thumb)

```

_main
        PUSH    {R4,R5,LR}
;
        SUB     SP, SP, #0x54
;

        MOVS   R0, #0           ; i
        MOV    R5, SP           ;

loc_1CE
        LSLS   R1, R0, #1       ; R1=i<<1 ( i*2)
        LSLS   R2, R0, #2       ; R2=i<<2 ( i*4)
        ADDS   R0, R0, #1       ; i=i+1

```

```

        CMP      R0, #20          ; i<20?
        STR      R1, [R5,R2]       ;
        BLT      loc_1CE         ;
;

        MOVS    R4, #0           ; i=0
loc_1DC
        LSLS    R0, R4, #2          ; R0=i<<2 ( i*4)
        LDR     R2, [R5,R0]        ; *(R5+R0) ( R5+i*4)
        MOVS    R1, R4
        ADR     R0, aADD          ; "a[%d]=%d\n"
        BL      __2printf
        ADDS    R4, R4, #1          ; i=i+1
        CMP     R4, #20          ; i<20?
        BLT      loc_1DC         ;
        MOVS    R0, #0           ;
;
        ADD     SP, SP, #0x54
        POP     {R4,R5,PC}

```

Sin optimización GCC 4.9.1 (ARM64)

Listing 18.3: Sin optimización GCC 4.9.1 (ARM64)

```

.LC0:
        .string "a[%d]=%d\n"
main:
; :
        stp      x29, x30, [sp, -112]!
; (FP=SP)
        add     x29, sp, 0
; :
        str     wzr, [x29,108]
; :
        b      .L2
.L3:
; :
        ldr     w0, [x29,108]
; 2:
        lsl     w2, w0, 1
; :
        add     x0, x29, 24
; :
        ldrsw   x1, [x29,108]
; (X0+X1<<2=array address+i*4) W2 (i*2) :

```

```

        str    w2, [x0,x1,lsl 2]
; (i):
        ldr    w0, [x29,108]
        add    w0, w0, 1
        str    w0, [x29,108]
.L2:
; :
        ldr    w0, [x29,108]
        cmp    w0, 19
; :
        ble   .L3
; .
; 0.
; .
        str    wzr, [x29,108]
        b     .L4
.L5:
; :
        add    x0, x29, 24
; :
        ldrsw  x1, [x29,108]
; (X0+X1<<2 = + i*4)
        ldr    w2, [x0,x1,lsl 2]
; "a[%d]=%d\n" :
        adrp   x0, .LC0
        add    x0, x0, :lo12:.LC0
; :
        ldr    w1, [x29,108]
; W2 .
; printf():
        b1    printf
; :
        ldr    w0, [x29,108]
        add    w0, w0, 1
        str    w0, [x29,108]
.L4:
; ?
        ldr    w0, [x29,108]
        cmp    w0, 19
; :
        ble   .L5
; 0
        mov    w0, 0
; :
        ldp    x29, x30, [sp], 112
        ret

```

18.1.3 MIPS

Listing 18.4: Con optimización GCC 4.4.5 (IDA)

```

main:
var_70      = -0x70
var_68      = -0x68
var_14      = -0x14
var_10      = -0x10
var_C       = -0xC
var_8       = -8
var_4       = -4
; :
        lui      $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x80
        la      $gp, (__gnu_local_gp & 0xFFFF)
        sw      $ra, 0x80+var_4($sp)
        sw      $s3, 0x80+var_8($sp)
        sw      $s2, 0x80+var_C($sp)
        sw      $s1, 0x80+var_10($sp)
        sw      $s0, 0x80+var_14($sp)
        sw      $gp, 0x80+var_70($sp)
        addiu   $s1, $sp, 0x80+var_68
        move    $v1, $s1
        move    $v0, $zero
; :
; :
        li      $a0, 0x28 # '('
loc_34:                                # CODE XREF: main+3C
; :
        sw      $v0, 0($v1)
; :
        addiu  $v0, 2
; ?
        bne   $v0, $a0, loc_34
; :
        addiu  $v1, 4
;
;
        la      $s3, $LC0      # "a[%d]=%d\n"
; $s0:
        move   $s0, $zero
        li      $s2, 0x14
loc_54:                                # CODE XREF: main+70

```

```

; printf():
    lw      $t9,  (printf & 0xFFFF)($gp)
    lw      $a2,  0($s1)
    move   $a1,  $s0
    move   $a0,  $s3
    jalr   $t9
; "i":
    addiu $s0,  1
    lw     $gp,  0x80+var_70($sp)
; :
    bne   $s0,  $s2,  loc_54
; :
    addiu $s1,  4
;
    lw     $ra,  0x80+var_4($sp)
    move   $v0,  $zero
    lw     $s3,  0x80+var_8($sp)
    lw     $s2,  0x80+var_C($sp)
    lw     $s1,  0x80+var_10($sp)
    lw     $s0,  0x80+var_14($sp)
    jr     $ra
    addiu $sp,  0x80
$LC0:    .ascii  "a[%d]=%d\n"<0>    # DATA XREF: main+44

```

[39 on page 620.](#)

18.2

18.2.1

```
#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<20; i++)
        a[i]=i*2;

    printf ("a[20]=%d\n", a[20]);

    return 0;
}
```

(MSVC 2008):

Listing 18.5: Sin optimización MSVC 2008

```
$SG2474 DB      'a[20]=%d', 0aH, 00H

_i$ = -84 ; size = 4
_a$ = -80 ; size = 80
_main    PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 84
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp     DWORD PTR _i$[ebp], 20
    jge     SHORT $LN1@main
    mov     ecx, DWORD PTR _i$[ebp]
    shl     ecx, 1
    mov     edx, DWORD PTR _i$[ebp]
    mov     DWORD PTR _a$[ebp+edx*4], ecx
    jmp     SHORT $LN2@main
$LN1@main:
    mov     eax, DWORD PTR _a$[ebp+80]
    push   eax
    push   OFFSET $SG2474 ; 'a[20]=%d'
    call   DWORD PTR __imp__printf
    add     esp, 8
    xor     eax, eax
    mov     esp, ebp
    pop     ebp
    ret     0
_main    ENDP
_TEXT   ENDS
END
```

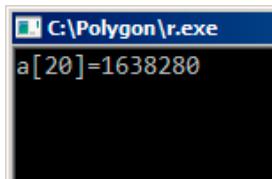


Figura 18.2: OllyDbg:

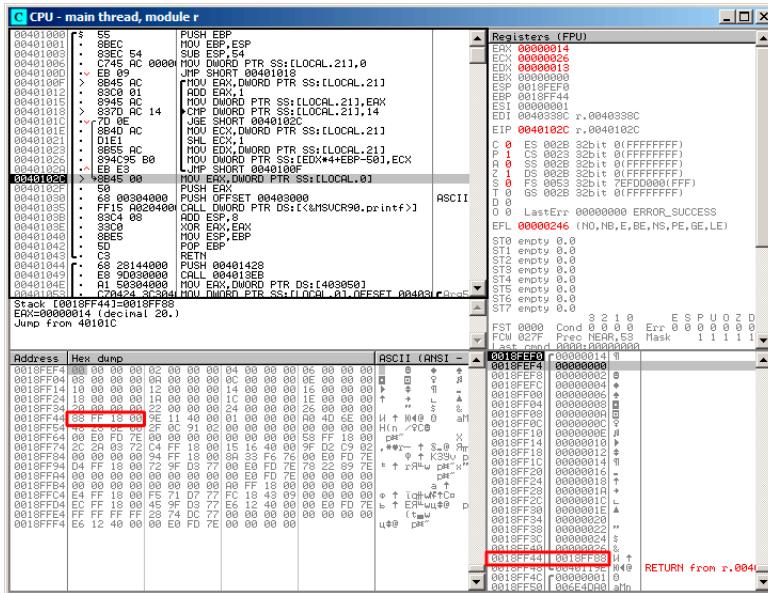


Figura 18.3: OllyDbg:

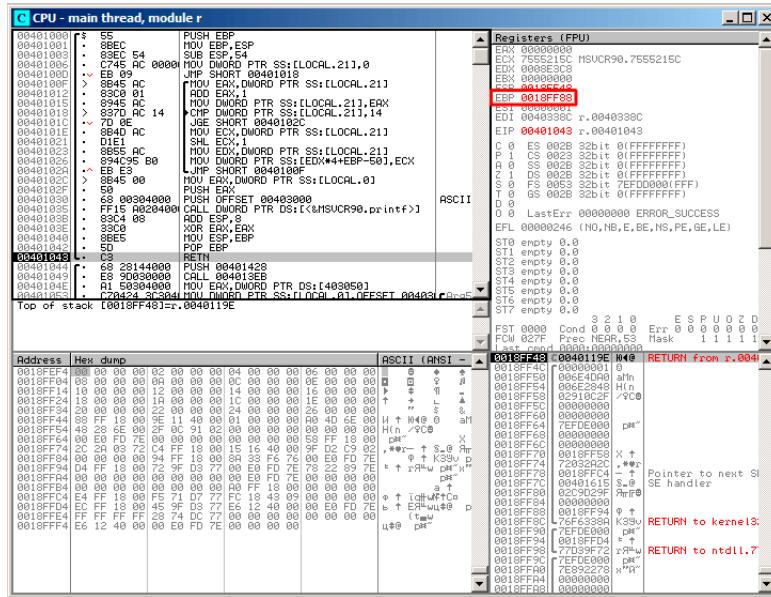


Figura 18.4: OllyDbg

18.2.2

```
#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<30; i++)
        a[i]=i;

    return 0;
}
```

MSVC

Listing 18.6: Sin optimización MSVC 2008

```
_TEXT      SEGMENT
```

```
_i$ = -84 ; size = 4
_a$ = -80 ; size = 80
_main PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 84
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp     DWORD PTR _i$[ebp], 30 ; 0000001eH
    jge     SHORT $LN1@main
    mov     ecx, DWORD PTR _i$[ebp]
    mov     edx, DWORD PTR _i$[ebp]          ;
    mov     DWORD PTR _a$[ebp+ecx*4], edx ;
    jmp     SHORT $LN2@main
$LN1@main:
    xor     eax, eax
    mov     esp, ebp
    pop     ebp
    ret     0
_main ENDP
```

OllyDbg,

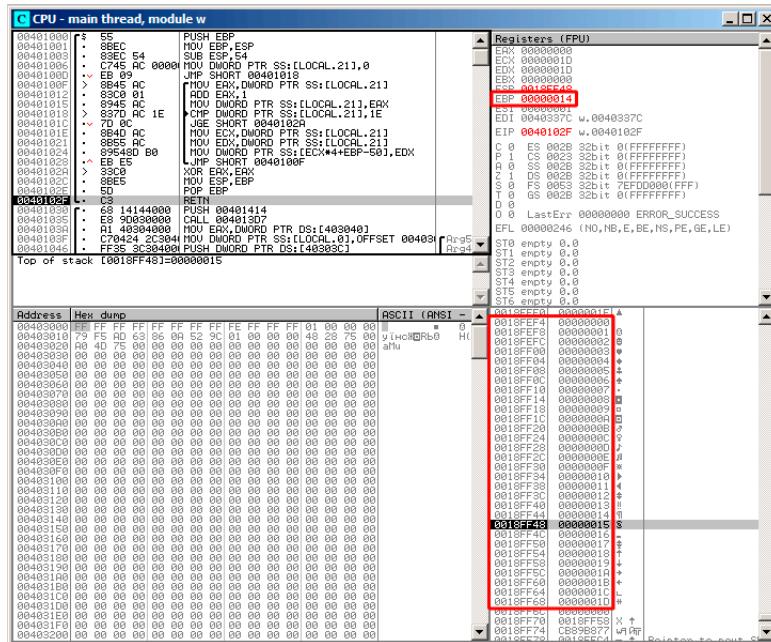


Figura 18.5: OllyDbg:

:

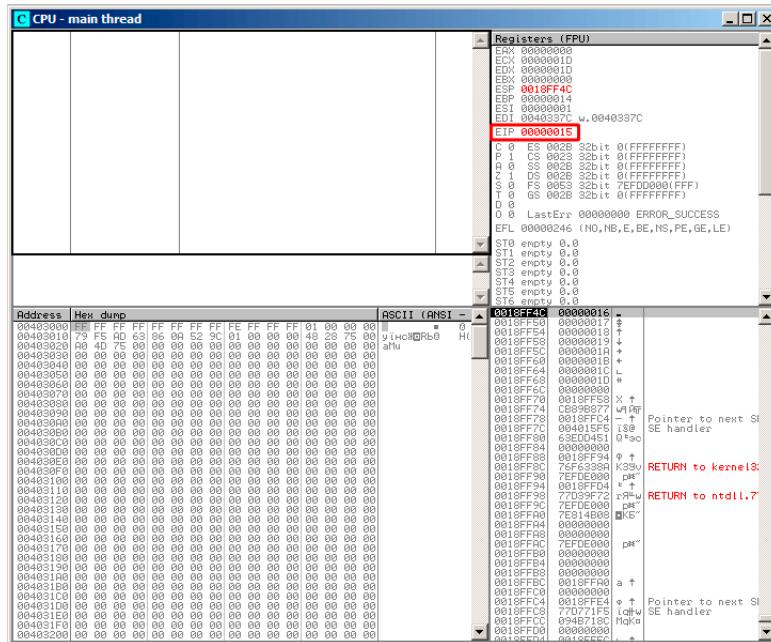


Figura 18.6: OllyDbg:

main():

ESP	
ESP+4	
ESP+84	
ESP+88	

buffer overflow¹.**GCC**

```

public main
proc near

a           = dword ptr -54h
i           = dword ptr -4

push      ebp

```

¹wikipedia

```

        mov    ebp, esp
        sub    esp, 60h ; 96
        mov    [ebp+i], 0
        jmp    short loc_80483D1
loc_80483C3:
        mov    eax, [ebp+i]
        mov    edx, [ebp+i]
        mov    [ebp+eax*4+a], edx
        add    [ebp+i], 1
loc_80483D1:
        cmp    [ebp+i], 1Dh
        jle    short loc_80483C3
        mov    eax, 0
        leave
        retn
main
        endp

```

Segmentation fault.

```

(gdb) r
Starting program: /home/dennis/RE/1

Program received signal SIGSEGV, Segmentation fault.
0x000000016 in ?? ()
(gdb) info registers
eax          0x0      0
ecx          0xd2f96388      -755407992
edx          0x1d      29
ebx          0x26efff4 2551796
esp          0xbfffff4b0      0xbfffff4b0
ebp          0x15      0x15
esi          0x0      0
edi          0x0      0
eip          0x16      0x16
eflags        0x10202  [ IF RF ]
cs           0x73      115
ss           0x7b      123
ds           0x7b      123
es           0x7b      123
fs           0x0      0
gs           0x33      51
(gdb)

```

18.3

[2.](#)

```
/RTCs Stack Frame runtime checking
/GZ Enable stack checks (/RTCs)
```

(18.1 on page 326) en [MSVC³](#), @_RTC_CheckStackVars@8

. alloca() (5.2.4 on page 36):

```
#ifdef __GNUC__
#include <alloca.h> // GCC
#else
#include <malloc.h> // MSVC
#endif
#include <stdio.h>

void f()
{
    char *buf=(char*)alloca (600);
#ifdef __GNUC__
    sprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // GCC
#else
    _snprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // MSVC
#endif

    puts (buf);
};
```

Listing 18.7: GCC 4.7.3

```
.LC0:
    .string "hi! %d, %d, %d\n"
f:
    push    ebp
    mov     ebp, esp
    push    ebx
    sub     esp, 676
    lea     ebx, [esp+39]
    and     ebx, -16
    mov     DWORD PTR [esp+20], 3
    mov     DWORD PTR [esp+16], 2
    mov     DWORD PTR [esp+12], 1
```

²: wikipedia.org/wiki/Buffer_overflow_protection

³Microsoft Visual C++

```

        mov     DWORD PTR [esp+8], OFFSET FLAT:.LC0 ; "hi! %d,%d\n"
        ↵ %d, %d\n"
        mov     DWORD PTR [esp+4], 600
        mov     DWORD PTR [esp], ebx
        mov     eax, DWORD PTR gs:20
        mov     DWORD PTR [ebp-12], eax
        xor     eax, eax
        call    _snprintf
        mov     DWORD PTR [esp], ebx
        call    puts
        mov     eax, DWORD PTR [ebp-12]
        xor     eax, DWORD PTR gs:20
        jne    .L5
        mov     ebx, DWORD PTR [ebp-4]
        leave
        ret
.L5:
        call    __stack_chk_fail

```

gs:20. gs:20. __stack_chk_fail (Ubuntu 13.04 x86):

```

*** buffer overflow detected ***: ./2_1 terminated
===== Backtrace: =====
/lib/i386-linux-gnu/libc.so.6(__fortify_fail+0x63)[0xb7699bc3]
/lib/i386-linux-gnu/libc.so.6(+0x10593a)[0xb769893a]
/lib/i386-linux-gnu/libc.so.6(+0x105008)[0xb7698008]
/lib/i386-linux-gnu/libc.so.6(_IO_default_xsputn+0x8c)[0xb7606e5c]
/lib/i386-linux-gnu/libc.so.6(_IO_vfprintf+0x165)[0xb75d7a45]
/lib/i386-linux-gnu/libc.so.6(__vsprintf_chk+0xc9)[0xb76980d9]
/lib/i386-linux-gnu/libc.so.6(__sprintf_chk+0x2f)[0xb7697fef]
./2_1[0x8048404]
/lib/i386-linux-gnu/libc.so.6(__libc_start_main+0xf5)[0xb75ac935]
===== Memory map: =====
08048000-08049000 r-xp 00000000 08:01 2097586 /home/dennis/2_1
    ↵ _1
08049000-0804a000 r--p 00000000 08:01 2097586 /home/dennis/2_1
    ↵ _1
0804a000-0804b000 rw-p 00001000 08:01 2097586 /home/dennis/2_1
    ↵ _1
094d1000-094f2000 rw-p 00000000 00:00 0 [heap]
b7560000-b757b000 r-xp 00000000 08:01 1048602 /lib/i386-
    ↵ linux-gnu/libgcc_s.so.1
b757b000-b757c000 r--p 0001a000 08:01 1048602 /lib/i386-
    ↵ linux-gnu/libgcc_s.so.1
b757c000-b757d000 rw-p 0001b000 08:01 1048602 /lib/i386-
    ↵ linux-gnu/libgcc_s.so.1

```

```
b7592000-b7593000 rw-p 00000000 00:00 0
b7593000-b7740000 r-xp 00000000 08:01 1050781      /lib/i386-
  ↳ linux-gnu/libc-2.17.so
b7740000-b7742000 r--p 001ad000 08:01 1050781      /lib/i386-
  ↳ linux-gnu/libc-2.17.so
b7742000-b7743000 rw-p 001af000 08:01 1050781      /lib/i386-
  ↳ linux-gnu/libc-2.17.so
b7743000-b7746000 rw-p 00000000 00:00 0
b775a000-b775d000 rw-p 00000000 00:00 0
b775d000-b775e000 r-xp 00000000 00:00 0          [vdso]
b775e000-b777e000 r-xp 00000000 08:01 1050794      /lib/i386-
  ↳ linux-gnu/ld-2.17.so
b777e000-b777f000 r--p 0001f000 08:01 1050794      /lib/i386-
  ↳ linux-gnu/ld-2.17.so
b777f000-b7780000 rw-p 00020000 08:01 1050794      /lib/i386-
  ↳ linux-gnu/ld-2.17.so
bff35000-bff56000 rw-p 00000000 00:00 0          [stack]
Aborted (core dumped)
```

gs TIB⁴ ⁵.

arch/x86/include/asm/stackprotector.h

18.3.1 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

(18.1 on page 326),

```
_main

var_64          = -0x64
var_60          = -0x60
var_5C          = -0x5C
var_58          = -0x58
var_54          = -0x54
var_50          = -0x50
var_4C          = -0x4C
var_48          = -0x48
var_44          = -0x44
var_40          = -0x40
var_3C          = -0x3C
var_38          = -0x38
var_34          = -0x34
var_30          = -0x30
var_2C          = -0x2C
```

⁴Thread Information Block

⁵wikipedia.org/wiki/Win32_Thread_Information_Block

```

var_28          = -0x28
var_24          = -0x24
var_20          = -0x20
var_1C          = -0x1C
var_18          = -0x18
canary          = -0x14
var_10          = -0x10

PUSH    {R4-R7,LR}
ADD     R7, SP, #0xC
STR.W   R8, [SP,#0xC+var_10]!
SUB     SP, SP, #0x54
MOVW   R0, #aObjc_methtype ; "objc_methtype"
MOVS   R2, #0
MOVT.W  R0, #0
MOVS   R5, #0
ADD    R0, PC
LDR.W  R8, [R0]
LDR.W  R0, [R8]
STR    R0, [SP,#0x64+canary]
MOVS   R0, #2
STR    R2, [SP,#0x64+var_64]
STR    R0, [SP,#0x64+var_60]
MOVS   R0, #4
STR    R0, [SP,#0x64+var_5C]
MOVS   R0, #6
STR    R0, [SP,#0x64+var_58]
MOVS   R0, #8
STR    R0, [SP,#0x64+var_54]
MOVS   R0, #0xA
STR    R0, [SP,#0x64+var_50]
MOVS   R0, #0xC
STR    R0, [SP,#0x64+var_4C]
MOVS   R0, #0xE
STR    R0, [SP,#0x64+var_48]
MOVS   R0, #0x10
STR    R0, [SP,#0x64+var_44]
MOVS   R0, #0x12
STR    R0, [SP,#0x64+var_40]
MOVS   R0, #0x14
STR    R0, [SP,#0x64+var_3C]
MOVS   R0, #0x16
STR    R0, [SP,#0x64+var_38]
MOVS   R0, #0x18
STR    R0, [SP,#0x64+var_34]
MOVS   R0, #0x1A
STR    R0, [SP,#0x64+var_30]

```

```

MOVS    R0, #0x1C
STR     R0, [SP,#0x64+var_2C]
MOVS    R0, #0x1E
STR     R0, [SP,#0x64+var_28]
MOVS    R0, #0x20
STR     R0, [SP,#0x64+var_24]
MOVS    R0, #0x22
STR     R0, [SP,#0x64+var_20]
MOVS    R0, #0x24
STR     R0, [SP,#0x64+var_1C]
MOVS    R0, #0x26
STR     R0, [SP,#0x64+var_18]
MOV     R4, 0xFDA ; "a[%d]=%d\n"
MOV     R0, SP
ADDS   R6, R0, #4
ADD    R4, PC
B      loc_2F1C

;

loc_2F14
ADDS   R0, R5, #1
LDR.W  R2, [R6,R5,LSL#2]
MOV    R5, R0

loc_2F1C
MOV    R0, R4
MOV    R1, R5
BLX    _printf
CMP    R5, #0x13
BNE   loc_2F14
LDR.W  R0, [R8]
LDR    R1, [SP,#0x64+canary]
CMP    R0, R1
ITTTT EQ          ; ?
MOVEQ  R0, #0
ADDEQ  SP, SP, #0x54
LDREQ.W R8, [SP+0x64+var_64],#4
POPEQ  {R4-R7,PC}
BLX    __stack_chk_fail

```

__stack_chk_fail

18.4

void f(int size)

```
{
    int a[size];
...
};
```

18.5

Listing 18.8:

```
#include <stdio.h>

const char* month1[]=
{
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
};

// 
const char* get_month1 (int month)
{
    return month1[month];
};
```

18.5.1 x64

Listing 18.9: Con optimización MSVC 2013 x64

_DATA	SEGMENT
month1	DQ FLAT:\$SG3122
	DQ FLAT:\$SG3123
	DQ FLAT:\$SG3124
	DQ FLAT:\$SG3125
	DQ FLAT:\$SG3126
	DQ FLAT:\$SG3127

```

DQ    FLAT:$SG3128
DQ    FLAT:$SG3129
DQ    FLAT:$SG3130
DQ    FLAT:$SG3131
DQ    FLAT:$SG3132
DQ    FLAT:$SG3133
$SG3122 DB  'January', 00H
$SG3123 DB  'February', 00H
$SG3124 DB  'March', 00H
$SG3125 DB  'April', 00H
$SG3126 DB  'May', 00H
$SG3127 DB  'June', 00H
$SG3128 DB  'July', 00H
$SG3129 DB  'August', 00H
$SG3130 DB  'September', 00H
$SG3156 DB  '%s', 0AH, 00H
$SG3131 DB  'October', 00H
$SG3132 DB  'November', 00H
$SG3133 DB  'December', 00H
_DATA    ENDS

month$ = 8
get_month1 PROC
    movsx   rax, ecx
    lea     rcx, OFFSET FLAT:month1
    mov     rax, QWORD PTR [rcx+rax*8]
    ret    0
get_month1 ENDP

```

:

- [6](#).
-
- .

Con optimización GCC 4.9 [7](#):

Listing 18.10: Con optimización GCC 4.9 x64

```

movsx  rdi, edi
mov    rax, QWORD PTR month1[0+rdi*8]
ret

```

6

7

32-bit MSVC

Listing 18.11: Con optimización MSVC 2013 x86

```
_month$ = 8
_get_month1 PROC
    mov     eax, DWORD PTR _month$[esp-4]
    mov     eax, DWORD PTR _month1[eax*4]
    ret     0
_get_month1 ENDP
```

18.5.2 32- ARM**ARM**

Listing 18.12: Con optimización Keil 6/2013 (Modo ARM)

```
get_month1 PROC
    LDR     r1, |L0.100|
    LDR     r0, [r1,r0,LSL #2]
    BX      lr
ENDP

|L0.100|
    DCD    || .data ||
    DCB    "January",0
    DCB    "February",0
    DCB    "March",0
    DCB    "April",0
    DCB    "May",0
    DCB    "June",0
    DCB    "July",0
    DCB    "August",0
    DCB    "September",0
    DCB    "October",0
    DCB    "November",0
    DCB    "December",0

    AREA  || .data ||, DATA, ALIGN=2
month1
    DCD    || .conststring ||
    DCD    || .conststring ||+0x8
    DCD    || .conststring ||+0x11
    DCD    || .conststring ||+0x17
    DCD    || .conststring ||+0x1d
    DCD    || .conststring ||+0x21
    DCD    || .conststring ||+0x26
```

```

    DCD    |||.conststring||+0x2b
    DCD    |||.conststring||+0x32
    DCD    |||.conststring||+0x3c
    DCD    |||.conststring||+0x44
    DCD    |||.conststring||+0x4d

```

ARM

```

get_month1 PROC
    LSLS    r0,r0,#2
    LDR    r1,|L0.64|
    LDR    r0,[r1,r0]
    BX    lr
ENDP

```

18.5.3 ARM64

Listing 18.13: Con optimización GCC 4.9 ARM64

```

get_month1:
    adrp    x1, .LANCHOR0
    add    x1, x1, :lo12:.LANCHOR0
    ldr    x0, [x1,w0,sxtw 3]
    ret

.LANCHOR0 = . + 0
.type   month1, %object
.size   month1, 96

month1:
    .xword  .LC2
    .xword  .LC3
    .xword  .LC4
    .xword  .LC5
    .xword  .LC6
    .xword  .LC7
    .xword  .LC8
    .xword  .LC9
    .xword  .LC10
    .xword  .LC11
    .xword  .LC12
    .xword  .LC13

.LC2:
    .string "January"
.LC3:
    .string "February"
.LC4:

```

```

        .string "March"
.LC5:    .string "April"
.LC6:    .string "May"
.LC7:    .string "June"
.LC8:    .string "July"
.LC9:    .string "August"
.LC10:   .string "September"
.LC11:   .string "October"
.LC12:   .string "November"
.LC13:   .string "December"

```

18.5.4 MIPS

Listing 18.14: Con optimización GCC 4.4.5 (IDA)

```

get_month1:
; $v0:
; 4:           la      $v0, month1
; :           sll     $a0, 2
; $v0:         addu   $a0, $v0
;           lw      $v0, 0($a0)
;
;           jr      $ra
;           or      $at, $zero ; branch delay slot, NOP

           .data # .data.rel.local
           .globl month1
month1:   .word aJanuary          # "January"
           .word aFebruary         # "February"
           .word aMarch            # "March"
           .word aApril             # "April"
           .word aMay               # "May"
           .word aJune              # "June"
           .word aJuly              # "July"

```

```

.word aAugust          # "August"
.word aSeptember      # "September"
.word aOctober         # "October"
.word aNovember        # "November"
.word aDecember        # "December"

.data # .rodata.str1.4
aJanuary:             .ascii "January"<0>
aFebruary:            .ascii "February"<0>
aMarch:               .ascii "March"<0>
aApril:               .ascii "April"<0>
aMay:                 .ascii "May"<0>
aJune:                .ascii "June"<0>
aJuly:                .ascii "July"<0>
aAugust:              .ascii "August"<0>
aSeptember:           .ascii "September"<0>
aOctober:              .ascii "October"<0>
aNoverember:           .ascii "Noverember"<0>
aDecember:             .ascii "December"<0>

```

18.5.5

Listing 18.15: IDA

```

off_140011000 dq offset aJanuary_1 ; DATA XREF: .text ↴
↳ :0000000140001003
                ; "January"
dq offset aFebruary_1 ; "February"
dq offset aMarch_1   ; "March"
dq offset aApril_1   ; "April"
dq offset aMay_1    ; "May"
dq offset aJune_1   ; "June"
dq offset aJuly_1   ; "July"
dq offset aAugust_1 ; "August"
dq offset aSeptember_1 ; "September"
dq offset aOctober_1 ; "October"
dq offset aNovember_1 ; "November"
dq offset aDecember_1 ; "December"
aJanuary_1 db 'January',0 ; DATA XREF: ↴
↳ sub_140001020+4 ; .data:off_140011000
aFebruary_1 db 'February',0 ; DATA XREF: .data ↴
↳ :0000000140011008 align 4
aMarch_1 db 'March',0 ; DATA XREF: .data ↴
↳ :0000000140011010 align 4

```

```
aApril_1      db 'April',0           ; DATA XREF: .data✓
↳ :0000000140011018
```

Listing 18.16: IDA

```
off_140011000 dq offset qword_140011060
                ; DATA XREF: .text✓
↳ :0000000140001003
    dq offset aFebruary_1   ; "February"
    dq offset aMarch_1     ; "March"
    dq offset aApril_1     ; "April"
    dq offset aMay_1       ; "May"
    dq offset aJune_1      ; "June"
    dq offset aJuly_1      ; "July"
    dq offset aAugust_1    ; "August"
    dq offset aSeptember_1 ; "September"
    dq offset aOctober_1   ; "October"
    dq offset aNovember_1  ; "November"
    dq offset aDecember_1  ; "December"
qword_140011060 dq 797261756E614Ah   ; DATA XREF: ✓
    ↳ sub_140001020+4
                ; .data:off_140011000
aFebruary_1    db 'February',0        ; DATA XREF: .data✓
    ↳ :0000000140011008
    align 4
aMarch_1       db 'March',0         ; DATA XREF: .data✓
    ↳ :0000000140011010
```

0x797261756E614A.

Si algo puede salir mal, saldrá mal

Ley de Murphy

Listing 18.17: assert()

```
const char* get_month1_checked (int month)
{
    assert (month<12);
    return month1[month];
}
```

Listing 18.18: Con optimización MSVC 2013 x64

```
$SG3143 DB      'm', 00H, 'o', 00H, 'n', 00H, 't', 00H, 'h', 00H
    ↳ H, '.', 00H
        DB      'c', 00H, 00H, 00H
$SG3144 DB      'm', 00H, 'o', 00H, 'n', 00H, 't', 00H, 'h', 00H
    ↳ H, '<', 00H
        DB      '1', 00H, '2', 00H, 00H, 00H

month$ = 48
get_month1_checked PROC
$LN5:
    push    rbx
    sub     rsp, 32
    movsxd  rbx, ecx
    cmp     ebx, 12
    jl      SHORT $LN3@get_month1
    lea     rdx, OFFSET FLAT:$SG3143
    lea     rcx, OFFSET FLAT:$SG3144
    mov     r8d, 29
    call    _wassert
$LN3@get_month1:
    lea     rcx, OFFSET FLAT:month1
    mov     rax, QWORD PTR [rcx+rbx*8]
    add     rsp, 32
    pop     rbx
    ret     0
get_month1_checked ENDP
```

.

:

Listing 18.19: Con optimización GCC 4.9 x64

```
.LC1:
    .string "month.c"
.LC2:
    .string "month<12"

get_month1_checked:
    cmp    edi, 11
    jg     .L6
    movsx  rdi, edi
    mov    rax, QWORD PTR month1[0+rdi*8]
    ret

.L6:
    push   rax
    mov    ecx, OFFSET FLAT:__PRETTY_FUNCTION__.2423
    mov    edx, 29
```

```

    mov    esi, OFFSET FLAT:.LC1
    mov    edi, OFFSET FLAT:.LC2
    call   __assert_fail

__PRETTY_FUNCTION__.2423:
.string "get_month1_checked"

```

8.

18.6

0	[0][0]
1	[0][1]
2	[0][2]
3	[0][3]
4	[1][0]
5	[1][1]
6	[1][2]
7	[1][3]
8	[2][0]
9	[2][1]
10	[2][2]
11	[2][3]

Cuadro 18.1:

0	1	2	3
4	5	6	7
8	9	10	11

Cuadro 18.2:

row-major order, C/C++ y Python. *row-major order*

column-major order () FORTRAN, MATLAB y R. *column-major order*

?

⁸ [msdn.microsoft.com/en-us/library/windows/hardware/ff543450\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff543450(v=vs.85).aspx)

18.6.1

0..3:

Listing 18.20:

```
#include <stdio.h>

char a[3][4];

int main()
{
    int x, y;

    //
    for (x=0; x<3; x++)
        for (y=0; y<4; y++)
            a[x][y]=0;

    // 0..3:
    for (y=0; y<4; y++)
        a[1][y]=y;
}
```

. 0, 1, 2 y 3:

Address	Hex dump
00C33370	00 00 00 00 00 01 02 03 00 00 00 00 00 00 00 00
00C33380	02 00 00 00 C3 66 47 4E C3 66 47 4E 00 00 00 00
00C33390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C333A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C333B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figura 18.7: OllyDbg:

0..2:

Listing 18.21:

```
#include <stdio.h>
```

```

char a[3][4];

int main()
{
    int x, y;

    //
    for (x=0; x<3; x++)
        for (y=0; y<4; y++)
            a[x][y]=0;

    // 0..2:
    for (x=0; x<3; x++)
        a[x][2]=x;
}

```

. 0, 1 y 2.

Address	Hex dump
01033380	00 00 00 00 00 00 01 00 00 00 02 00 02
01033390	00 00 00 00 1E AA EF 31 1E AA EF 31 00
010333A0	00 00 00 00 00 00 00 00 00 00 00 00 00
010333B0	00 00 00 00 00 00 00 00 00 00 00 00 00

Figura 18.8: OllyDbg:

18.6.2

```

#include <stdio.h>

char a[3][4];

char get_by_coordinates1 (char array[3][4], int a, int b)
{
    return array[a][b];
};

char get_by_coordinates2 (char *array, int a, int b)
{
    //
    // 4
    return array[a*4+b];
};

```

```

char get_by_coordinates3 (char *array, int a, int b)
{
    //
    //
    // 4
    return *(array+a*4+b);
};

int main()
{
    a[2][3]=123;
    printf ("%d\n", get_by_coordinates1(a, 2, 3));
    printf ("%d\n", get_by_coordinates2(a, 2, 3));
    printf ("%d\n", get_by_coordinates3(a, 2, 3));
}

```

Listing 18.22: Con optimización MSVC 2013 x64

```

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates3 PROC
; RCX=
; RDX=a
; R8=b
        movsxd    rax, r8d
; EAX=b
        movsxd    r9, edx
; R9=a
        add       rax, rcx
; RAX=b+
        movzx    eax, BYTE PTR [rax+r9*4]
; AL= RAX+R9*4=b++a*4=+a*4+b
        ret       0
get_by_coordinates3 ENDP

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates2 PROC
        movsxd    rax, r8d
        movsxd    r9, edx
        add       rax, rcx
        movzx    eax, BYTE PTR [rax+r9*4]
        ret       0
get_by_coordinates2 ENDP

```

```

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates1 PROC
    movsxd  rax, r8d
    movsxd  r9, edx
    add     rax, rcx
    movzx   eax, BYTE PTR [rax+r9*4]
    ret     0
get_by_coordinates1 ENDP

```

Listing 18.23: Con optimización GCC 4.9 x64

```

; RDI=
; RSI=a
; RDX=b

get_by_coordinates1:
;
    movsx   rsi, esi
    movsx   rdx, edx
    lea     rax, [rdi+rsi*4]
; RAX=RDI+RSI*4+=a*4
    movzx   eax, BYTE PTR [rax+rdx]
; AL= RAX+RDX+=a*4+b
    ret

get_by_coordinates2:
    lea     eax, [rdx+rsi*4]
; RAX=RDX+RSI*4=b+a*4
    cdqe
    movzx   eax, BYTE PTR [rdi+rax]
; AL= RDI+RAX+=b+a*4
    ret

get_by_coordinates3:
    sal     esi, 2
; ESI=a<<2=a*4
;
    movsx   rdx, edx
    movsx   rsi, esi
    add     rdi, rsi
; RDI=RDI+RSI+=a*4
    movzx   eax, BYTE PTR [rdi+rdx]
; AL= RDI+RDX+=a*4+b
    ret

```

18.6.3

:

Listing 18.24:

```
#include <stdio.h>

int a[10][20][30];

void insert(int x, int y, int z, int value)
{
    a[x][y][z]=value;
}
```

x86

(MSVC 2010):

Listing 18.25: MSVC 2010

```
_DATA      SEGMENT
COMM       _a:DWORD:01770H
_DATA      ENDS
PUBLIC     _insert
_TEXT      SEGMENT
_x$ = 8          ; size = 4
_y$ = 12         ; size = 4
_z$ = 16         ; size = 4
_value$ = 20      ; size = 4
_insert      PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _x$[ebp]
    imul   eax, 2400           ; eax=600*4*x
    mov     ecx, DWORD PTR _y$[ebp]
    imul   ecx, 120            ; ecx=30*4*y
    lea    edx, DWORD PTR _a[eax+ecx] ; edx=a + 600*4*x + 
    \ 30*4*y
    mov     eax, DWORD PTR _z$[ebp]
    mov     ecx, DWORD PTR _value$[ebp]
    mov     DWORD PTR [edx+eax*4], ecx ; *(edx+z*4)=
    pop    ebp
    ret    0
_insert      ENDP
_TEXT      ENDS
```

Listing 18.26: GCC 4.4.1

```

insert          public insert
                proc near

x              = dword ptr  8
y              = dword ptr 0Ch
z              = dword ptr 10h
value         = dword ptr 14h

                push    ebp
                mov     ebp, esp
                push    ebx
                mov     ebx, [ebp+x]
                mov     eax, [ebp+y]
                mov     ecx, [ebp+z]
                lea     edx, [eax+eax]           ; edx=y*2
                mov     eax, edx             ; eax=y*2
                shl     eax, 4              ; eax=(y*2)*
                ↳ <<4 = y*2*16 = y*32
                sub     eax, edx           ; eax=y*32 *
                ↳ - y*2=y*30
                imul   edx, ebx, 600        ; edx=x*600
                add    eax, edx           ; eax=eax+
                ↳ edx=y*30 + x*600
                lea     edx, [eax+ecx]       ; edx=y*30 *
                ↳ + x*600 + z
                mov     eax, [ebp+value]
                mov     dword ptr ds:a[edx*4], eax ; *(a+edx*
                ↳ *4)=
                pop    ebx
                pop    ebp
                retn
insert         endp

```

$$\therefore : (y + y) \ll 4 - (y + y) = (2y) \ll 4 - 2y = 2 \cdot 16 \cdot y - 2y = 32y - 2y = 30y. \ .$$

ARM + Sin optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

Listing 18.27: Sin optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

```

_insert
value    = -0x10
z        = -0xC
y        = -8

```

```

x      = -4

;
SUB    SP, SP, #0x10
MOV    R9, 0xFC2 ; a
ADD    R9, PC
LDR.W R9, [R9]
STR    R0, [SP,#0x10+x]
STR    R1, [SP,#0x10+y]
STR    R2, [SP,#0x10+z]
STR    R3, [SP,#0x10+value]
LDR    R0, [SP,#0x10+value]
LDR    R1, [SP,#0x10+z]
LDR    R2, [SP,#0x10+y]
LDR    R3, [SP,#0x10+x]
MOV    R12, 2400
MUL.W R3, R3, R12
ADD    R3, R9
MOV    R9, 120
MUL.W R2, R2, R9
ADD    R2, R3
LSLS   R1, R1, #2 ; R1=R1<<2
ADD    R1, R2
STR    R0, [R1] ; R1 -
;
ADD    SP, SP, #0x10
BX    LR

```

Sin optimización LLVM

ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

Listing 18.28: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb)

```

_insert
MOVW   R9, #0x10FC
MOV.W  R12, #2400
MOVT.W R9, #0
RSB.W  R1, R1, R1, LSL#4 ; R1 - y. R1=y<<4 - y = y*16 - y = y*
    ↴ *15
ADD    R9, PC          ; R9 =
LDR.W  R9, [R9]
MLA.W  R0, R0, R12, R9 ; R0 - x, R12 - 2400, R9 - . R0=x*
    ↴ *2400 +
ADD.W  R0, R0, R1, LSL#3 ; R0 = R0+R1<<3 = R0+R1*8 = x*2400 + *
    ↴ + y*15*8 =

```

```

STR.W    R3, [R0,R2,LSL#2] ; R2 - z, R3 - . =R0+z*4 =
; + y*30*4 + x*600*4 +
; + y*30*4 + x*600*4 + z*4
BX      LR

```

RSB (*Reverse Subtract*). SUB y RSB: (LSL#4).

LDR.W R9, [R9] LEA ([A.6.2 on page 1233](#))

MIPS

Listing 18.29: Con optimización GCC 4.4.5 (IDA)

```

insert:
; $a0=x
; $a1=y
; $a2=z
; $a3=
           sll      $v0, $a0, 5
; $v0 = $a0<<5 = x*32
           sll      $a0, 3
; $a0 = $a0<<3 = x*8
           addu   $a0, $v0
; $a0 = $a0+$v0 = x*8+x*32 = x*40
           sll      $v1, $a1, 5
; $v1 = $a1<<5 = y*32
           sll      $v0, $a0, 4
; $v0 = $a0<<4 = x*40*16 = x*640
           sll      $a1, 1
; $a1 = $a1<<1 = y*2
           subu   $a1, $v1, $a1
; $a1 = $v1-$a1 = y*32-y*2 = y*30
           subu   $a0, $v0, $a0
; $a0 = $v0-$a0 = x*640-x*40 = x*600
           la      $gp, __gnu_local_gp
           addu   $a0, $a1, $a0
; $a0 = $a1+$a0 = y*30+x*600
           addu   $a0, $a2
; $a0 = $a0+$a2 = y*30+x*600+z
; :
           lw      $v0, (a & 0xFFFF)($gp)
; :
           sll      $a0, 2
; :
           addu   $a0, $v0, $a0
; :
           jr      $ra

```

		SW \$a3, 0(\$a0)
--	--	------------------

		.comm a:0x1770
--	--	----------------

18.6.4

[.: 83.2 on page 1095.](#)

18.7

Spanish text placeholder.[18.8.](#)

```
#include <stdio.h>
#include <assert.h>

const char month2[12][10]=
{
    { 'J', 'a', 'n', 'u', 'a', 'r', 'y', 0, 0, 0 },
    { 'F', 'e', 'b', 'r', 'u', 'a', 'r', 'y', 0, 0 },
    { 'M', 'a', 'r', 'c', 'h', 0, 0, 0, 0, 0 },
    { 'A', 'p', 'r', 'i', 'l', 0, 0, 0, 0, 0 },
    { 'M', 'a', 'y', 0, 0, 0, 0, 0, 0, 0 },
    { 'J', 'u', 'n', 'e', 0, 0, 0, 0, 0, 0 },
    { 'J', 'u', 'l', 'y', 0, 0, 0, 0, 0, 0 },
    { 'A', 'u', 'g', 'u', 's', 't', 0, 0, 0, 0 },
    { 'S', 'e', 'p', 't', 'e', 'm', 'b', 'e', 'r', 0 },
    { 'O', 'c', 't', 'o', 'b', 'e', 'r', 0, 0, 0 },
    { 'N', 'o', 'v', 'e', 'm', 'b', 'e', 'r', 0, 0 },
    { 'D', 'e', 'c', 'e', 'm', 'b', 'e', 'r', 0, 0 }
};

// const char* get_month2 (int month)
{
    return &month2[month][0];
}
```

Listing 18.30: Con optimización MSVC 2013 x64

month2	DB	04aH
	DB	061H
	DB	06eH
	DB	075H

```

        DB      061H
        DB      072H
        DB      079H
        DB      00H
        DB      00H
        DB      00H
...
get_month2 PROC
;
    movsxsd  rax,  ecx
    lea     rcx,  QWORD PTR [rax+rax*4]
; RCX=+*4=*5
    lea     rax,  OFFSET FLAT:month2
; RAX=
    lea     rax,  QWORD PTR [rax+rcx*2]
; RAX= + RCX*2= + *5*2= + *10
    ret     0
get_month2 ENDP

```

. *pointer_to_the_table + month * 10.*

.

Con optimización GCC 4.9 :

Listing 18.31: Con optimización GCC 4.9 x64

```

movsx   rdi,  edi
lea     rax,  [rdi+rdi*4]
lea     rax,  month2[rax+rax]
ret

```

Listing 18.32: Sin optimización GCC 4.9 x64

```

get_month2:
    push   rbp
    mov    rbp,  rsp
    mov    DWORD PTR [rbp-4], edi
    mov    eax, DWORD PTR [rbp-4]
    movsx   rdx,  eax
; RDX =
    mov    rax,  rdx
; RAX =
    sal    rax,  2
; RAX = <<2 = *4
    add    rax,  rdx
; RAX = RAX+RDX = *4+ = *5

```

```

        add    rax, rax
; RAX = RAX*2 = *5*2 = *10
        add    rax, OFFSET FLAT:month2
; RAX = *10 +
        pop   rbp
        ret

```

Sin optimización MSVC :

Listing 18.33: Sin optimización MSVC 2013 x64

```

month$ = 8
get_month2 PROC
        mov    DWORD PTR [rsp+8], ecx
        movsxd rax, DWORD PTR month$[rsp]
; RAX =
        imul   rax, rax, 10
; RAX = RAX*10
        lea    rcx, OFFSET FLAT:month2
; RCX =
        add    rcx, rax
; RCX = RCX+RAX = +month*10
        mov    rax, rcx
; RAX = +*10
        mov    ecx, 1
; RCX = 1
        imul   rcx, rcx, 0
; RCX = 1*0 = 0
        add    rax, rcx
; RAX = +*10 + 0 = +*10
        ret    0
get_month2 ENDP

```

18.7.1 32-bit ARM

Con optimización Keil MULS:

Listing 18.34: Con optimización Keil 6/2013 (Modo Thumb)

```

; R0 =
        MOVS    r1,#0xa
; R1 = 10
        MULS    r0,r1,r0
; R0 = R1*R0 = 10*
        LDR     r1,|L0.68|

```

```
; R1 =
      ADDS      r0,r0,r1
; R0 = R0+R1 = 10* +
      BX       lr
```

Con optimización Keil :

Listing 18.35: Con optimización Keil 6/2013 (Modo ARM)

```
; R0 =
      LDR      r1, |L0.104|
; R1 =
      ADD      r0,r0,r0,LSL #2
; R0 = R0+R0<<2 = R0+R0*4 = *5
      ADD      r0,r1,r0,LSL #1
; R0 = R1+R0<<2 = + *5*2 = + *10
      BX       lr
```

18.7.2 ARM64

Listing 18.36: Con optimización GCC 4.9 ARM64

```
; W0 =
      sxtw    x0, w0
; X0 =
      adrp    x1, .LANCHOR1
      add     x1, x1, :lo12:.LANCHOR1
; X1 =
      add     x0, x0, x0, lsl 2
; X0 = X0+X0<<2 = X0+X0*4 = X0*5
      add     x0, x1, x0, lsl 1
; X0 = X1+X0<<1 = X1+X0*2 = + X0*10
      ret
```

18.7.3 MIPS

Listing 18.37: Con optimización GCC 4.4.5 (IDA)

```
.globl get_month2
get_month2:
; $a0=
      sll      $v0, $a0, 3
; $v0 = $a0<<3 = *8
```

```

        sll      $a0, 1
; $a0 = $a0<<1 = *2
        addu    $a0, $v0
; $a0 = *2+*8 = *10
; :
        la       $v0, month2
; :
        jr       $ra
        addu    $v0, $a0

month2:      .ascii "January"<0>
              .byte 0, 0
aFebruary:   .ascii "February"<0>
              .byte 0
aMarch:      .ascii "March"<0>
              .byte 0, 0, 0, 0
aApril:      .ascii "April"<0>
              .byte 0, 0, 0, 0
aMay:        .ascii "May"<0>
              .byte 0, 0, 0, 0, 0
aJune:       .ascii "June"<0>
              .byte 0, 0, 0, 0, 0
aJuly:        .ascii "July"<0>
              .byte 0, 0, 0, 0, 0
aAugust:     .ascii "August"<0>
              .byte 0, 0, 0
aSeptember:  .ascii "September"<0>
aOctober:    .ascii "October"<0>
              .byte 0, 0
aNovember:   .ascii "November"<0>
              .byte 0
aDecember:   .ascii "December"<0>
              .byte 0, 0, 0, 0, 0, 0, 0, 0, 0

```

18.7.4 Conclusión

18.8 Conclusión

18.9 Ejercicios

18.9.1 Ejercicio #1

Lo que hace el código?

Listing 18.38: MSVC 2010 + /O1

```

_a$ = 8          ; size = 4
_b$ = 12         ; size = 4
_c$ = 16         ; size = 4
?s@@YAXPAN00@Z PROC      ; s, COMDAT
    mov    eax, DWORD PTR _b$[esp-4]
    mov    ecx, DWORD PTR _a$[esp-4]
    mov    edx, DWORD PTR _c$[esp-4]
    push   esi
    push   edi
    sub    ecx, eax
    sub    edx, eax
    mov    edi, 200      ; 000000c8H
$LL6@s:
    push   100        ; 00000064H
    pop    esi
$LL3@s:
    fld    QWORD PTR [ecx+eax]
    fadd  QWORD PTR [eax]
    fstp  QWORD PTR [edx+eax]
    add    eax, 8
    dec    esi
    jne    SHORT $LL3@s
    dec    edi
    jne    SHORT $LL6@s
    pop    edi
    pop    esi
    ret    0
?s@@YAXPAN00@Z ENDP      ; s

```

(/O1:).

Listing 18.39: Con optimización Keil 6/2013 (Modo ARM)

	PUSH {r4-r12,lr}
	MOV r9,r2
	MOV r10,r1
	MOV r11,r0
	MOV r5,#0
L0.20	ADD r0,r5,r5,LSL #3
	ADD r0,r0,r5,LSL #4
	MOV r4,#0
	ADD r8,r10,r0,LSL #5
	ADD r7,r11,r0,LSL #5
	ADD r6,r9,r0,LSL #5
L0.44	ADD r0,r8,r4,LSL #3
	LDM r0,{r2,r3}

```

ADD    r1,r7,r4,LSL #3
LDM    r1,{r0,r1}
BL     __aeabi_dadd
ADD    r2,r6,r4,LSL #3
ADD    r4,r4,#1
STM    r2,{r0,r1}
CMP    r4,#0x64
BLT   |L0.44|
ADD    r5,r5,#1
CMP    r5,#0xc8
BLT   |L0.20|
POP   {r4-r12,pc}

```

Listing 18.40: Con optimización Keil 6/2013 (Modo Thumb)

```

PUSH   {r0-r2,r4-r7,lr}
MOVS   r4,#0
SUB   sp,sp,#8
|L0.6|
MOVS   r1,#0x19
MOVS   r0,r4
LSLS   r1,r1,#5
MULS   r0,r1,r0
LDR    r2,[sp,#8]
LDR    r1,[sp,#0xc]
ADDS   r2,r0,r2
STR    r2,[sp,#0]
LDR    r2,[sp,#0x10]
MOVS   r5,#0
ADDS   r7,r0,r2
ADDS   r0,r0,r1
STR    r0,[sp,#4]
|L0.32|
LSLS   r6,r5,#3
ADDS   r0,r0,r6
LDM    r0!,{r2,r3}
LDR    r0,[sp,#0]
ADDS   r1,r0,r6
LDM    r1,{r0,r1}
BL     __aeabi_dadd
ADDS   r2,r7,r6
ADDS   r5,r5,#1
STM    r2!,{r0,r1}
CMP    r5,#0x64
BGE   |L0.62|
LDR    r0,[sp,#4]
B      |L0.32|
|L0.62|

```

```

ADD$    r4,r4,#1
CMP     r4,#0xc8
BLT     |L0.6|
ADD     sp,sp,#0x14
POP    {r4-r7,pc}

```

Listing 18.41: Sin optimización GCC 4.9 (ARM64)

```

S:
    sub    sp, sp, #48
    str    x0, [sp,24]
    str    x1, [sp,16]
    str    x2, [sp,8]
    str    wzr, [sp,44]
    b     .L2

.L5:
    str    wzr, [sp,40]
    b     .L3

.L4:
    ldr    w1, [sp,44]
    mov    w0, 100
    mul    w0, w1, w0
    sxtw  x1, w0
    ldrsw x0, [sp,40]
    add    x0, x1, x0
    lsl    x0, x0, 3
    ldr    x1, [sp,8]
    add    x0, x1, x0
    ldr    w2, [sp,44]
    mov    w1, 100
    mul    w1, w2, w1
    sxtw  x2, w1
    ldrsw x1, [sp,40]
    add    x1, x2, x1
    lsl    x1, x1, 3
    ldr    x2, [sp,24]
    add    x1, x2, x1
    ldr    x2, [x1]
    ldr    w3, [sp,44]
    mov    w1, 100
    mul    w1, w3, w1
    sxtw  x3, w1
    ldrsw x1, [sp,40]
    add    x1, x3, x1
    lsl    x1, x1, 3
    ldr    x3, [sp,16]
    add    x1, x3, x1
    ldr    x1, [x1]

```

```

    fmov    d0, x2
    fmov    d1, x1
    fadd    d0, d0, d1
    fmov    x1, d0
    str     x1, [x0]
    ldr     w0, [sp,40]
    add    w0, w0, 1
    str     w0, [sp,40]
.L3:
    ldr     w0, [sp,40]
    cmp     w0, 99
    ble     .L4
    ldr     w0, [sp,44]
    add    w0, w0, 1
    str     w0, [sp,44]
.L2:
    ldr     w0, [sp,44]
    cmp     w0, 199
    ble     .L5
    add    sp, sp, 48
    ret

```

Listing 18.42: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

sub_0:
    li      $t3, 0x27100
    move   $t2, $zero
    li      $t1, 0x64 # 'd'

loc_10:                                # CODE XREF: sub_0+54
    addu   $t0, $a1, $t2
    addu   $a3, $a2, $t2
    move   $v1, $a0
    move   $v0, $zero

loc_20:                                # CODE XREF: sub_0+48
    lwc1   $f2, 4($v1)
    lwc1   $f0, 4($t0)
    lwc1   $f3, 0($v1)
    lwc1   $f1, 0($t0)
    addiu  $v0, 1
    add.d  $f0, $f2, $f0
    addiu  $v1, 8
    swc1   $f0, 4($a3)
    swc1   $f1, 0($a3)
    addiu  $t0, 8
    bne    $v0, $t1, loc_20
    addiu  $a3, 8

```

```

addiu    $t2, 0x320
bne     $t2, $t3, loc_10
addiu    $a0, 0x320
jr      $ra
or      $at, $zero

```

Responda [G.1.12 on page 1263](#).

18.9.2 Ejercicio #2

Lo que hace el código?

Listing 18.43: MSVC 2010 + /01

```

tv315 = -8           ; size = 4
tv291 = -4           ; size = 4
_a$ = 8              ; size = 4
_b$ = 12             ; size = 4
_c$ = 16             ; size = 4
?m@@YAXPAN00@Z PROC   ; m, COMDAT
    push    ebp
    mov     ebp, esp
    push    ecx
    push    ecx
    mov     edx, DWORD PTR _a$[ebp]
    push    ebx
    mov     ebx, DWORD PTR _c$[ebp]
    push    esi
    mov     esi, DWORD PTR _b$[ebp]
    sub     edx, esi
    push    edi
    sub     esi, ebx
    mov     DWORD PTR tv315[ebp], 100 ; 00000064H
$LL9@m:
    mov     eax, ebx
    mov     DWORD PTR tv291[ebp], 300 ; 0000012CH
$LL6@m:
    fldz
    lea     ecx, DWORD PTR [esi+eax]
    fstp   QWORD PTR [eax]
    mov     edi, 200                 ; 000000c8H
$LL3@m:
    dec     edi
    fld     QWORD PTR [ecx+edx]
    fmul   QWORD PTR [ecx]
    fadd   QWORD PTR [eax]
    fstp   QWORD PTR [eax]

```

```

jne    HORT $LL3@m
add    eax, 8
dec    DWORD PTR tv291[ebp]
jne    SHORT $LL6@m
add    ebx, 800           ; 00000320H
dec    DWORD PTR tv315[ebp]
jne    SHORT $LL9@m
pop    edi
pop    esi
pop    ebx
leave
ret    0
?m@@YAXPAN00@Z ENDP          ; m

```

(/O1:).

Listing 18.44: Con optimización Keil 6/2013 (Modo ARM)

```

PUSH   {r0-r2,r4-r11,lr}
SUB    sp,sp,#8
MOV    r5,#0
|L0.12|
LDR    r1,[sp,#0xc]
ADD    r0,r5,r5,LSL #3
ADD    r0,r0,r5,LSL #4
ADD    r1,r1,r0,LSL #5
STR    r1,[sp,#0]
LDR    r1,[sp,#8]
MOV    r4,#0
ADD    r11,r1,r0,LSL #5
LDR    r1,[sp,#0x10]
ADD    r10,r1,r0,LSL #5
|L0.52|
MOV    r0,#0
MOV    r1,r0
ADD    r7,r10,r4,LSL #3
STM    r7,{r0,r1}
MOV    r6,r0
LDR    r0,[sp,#0]
ADD    r8,r11,r4,LSL #3
ADD    r9,r0,r4,LSL #3
|L0.84|
LDM    r9,{r2,r3}
LDM    r8,{r0,r1}
BL     __aeabi_dmul
LDM    r7,{r2,r3}
BL     __aeabi_dadd
ADD    r6,r6,#1

```

```

STM    r7,{r0,r1}
CMP    r6,#0xc8
BLT    |L0.84|
ADD    r4,r4,#1
CMP    r4,#0x12c
BLT    |L0.52|
ADD    r5,r5,#1
CMP    r5,#0x64
BLT    |L0.12|
ADD    sp,sp,#0x14
POP   {r4-r11,pc}

```

Listing 18.45: Con optimización Keil 6/2013 (Modo Thumb)

```

PUSH   {r0-r2,r4-r7,lr}
MOVS   r0,#0
SUB   sp,sp,#0x10
STR   r0,[sp,#0]
|L0.8|
MOVS   r1,#0x19
LSLS   r1,r1,#5
MULS   r0,r1,r0
LDR    r2,[sp,#0x10]
LDR    r1,[sp,#0x14]
ADDS   r2,r0,r2
STR    r2,[sp,#4]
LDR    r2,[sp,#0x18]
MOVS   r5,#0
ADDS   r7,r0,r2
ADDS   r0,r0,r1
STR    r0,[sp,#8]
|L0.32|
LSLS   r4,r5,#3
MOVS   r0,#0
ADDS   r2,r7,r4
STR    r0,[r2,#0]
MOVS   r6,r0
STR    r0,[r2,#4]
|L0.44|
LDR    r0,[sp,#8]
ADDS   r0,r0,r4
LDM    r0!,{r2,r3}
LDR    r0,[sp,#4]
ADDS   r1,r0,r4
LDM    r1,{r0,r1}
BL     __aeabi_dmull
ADDS   r3,r7,r4
LDM    r3,{r2,r3}

```

```

BL      __aeabi_dadd
ADD$    r2,r7,r4
ADD$    r6,r6,#1
STM    r2!,{r0,r1}
CMP    r6,#0xc8
BLT    |L0.44|
MOVS   r0,#0xff
ADD$    r5,r5,#1
ADD$    r0,r0,#0x2d
CMP    r5,r0
BLT    |L0.32|
LDR    r0,[sp,#0]
ADD$    r0,r0,#1
CMP    r0,#0x64
STR    r0,[sp,#0]
BLT    |L0.8|
ADD    sp,sp,#0x1c
POP    {r4-r7,pc}

```

Listing 18.46: Sin optimización GCC 4.9 (ARM64)

```

m:
    sub    sp, sp, #48
    str    x0, [sp,24]
    str    x1, [sp,16]
    str    x2, [sp,8]
    str    wzr, [sp,44]
    b     .L2

.L7:
    str    wzr, [sp,40]
    b     .L3

.L6:
    ldr    w1, [sp,44]
    mov    w0, 100
    mul    w0, w1, w0
    sxtw  x1, w0
    ldrsw x0, [sp,40]
    add    x0, x1, x0
    lsl    x0, x0, 3
    ldr    x1, [sp,8]
    add    x0, x1, x0
    ldr    x1, .LC0
    str    x1, [x0]
    str    wzr, [sp,36]
    b     .L4

.L5:
    ldr    w1, [sp,44]
    mov    w0, 100

```

```
mul      w0, w1, w0
sxtw    x1, w0
ldrsw   x0, [sp,40]
add     x0, x1, x0
lsl     x0, x0, 3
ldr     x1, [sp,8]
add     x0, x1, x0
ldr     w2, [sp,44]
mov     w1, 100
mul     w1, w2, w1
sxtw    x2, w1
ldrsw   x1, [sp,40]
add     x1, x2, x1
lsl     x1, x1, 3
ldr     x2, [sp,8]
add     x1, x2, x1
ldr     x2, [x1]
ldr     w3, [sp,44]
mov     w1, 100
mul     w1, w3, w1
sxtw    x3, w1
ldrsw   x1, [sp,40]
add     x1, x3, x1
lsl     x1, x1, 3
ldr     x3, [sp,24]
add     x1, x3, x1
ldr     x3, [x1]
ldr     w4, [sp,44]
mov     w1, 100
mul     w1, w4, w1
sxtw    x4, w1
ldrsw   x1, [sp,40]
add     x1, x4, x1
lsl     x1, x1, 3
ldr     x4, [sp,16]
add     x1, x4, x1
ldr     x1, [x1]
fmov    d0, x3
fmov    d1, x1
fmul   d0, d0, d1
fmov   x1, d0
fmov   d0, x2
fmov   d1, x1
fadd   d0, d0, d1
fmov   x1, d0
str    x1, [x0]
ldr    w0, [sp,36]
```

```

        add    w0, w0, 1
        str    w0, [sp,36]
.L4:
        ldr    w0, [sp,36]
        cmp    w0, 199
        ble    .L5
        ldr    w0, [sp,40]
        add    w0, w0, 1
        str    w0, [sp,40]
.L3:
        ldr    w0, [sp,40]
        cmp    w0, 299
        ble    .L6
        ldr    w0, [sp,44]
        add    w0, w0, 1
        str    w0, [sp,44]
.L2:
        ldr    w0, [sp,44]
        cmp    w0, 99
        ble    .L7
        add    sp, sp, 48
        ret

```

Listing 18.47: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

m:
        li     $t5, 0x13880
        move   $t4, $zero
        li     $t1, 0xC8
        li     $t3, 0x12C

loc_14:                                # CODE XREF: m+7C
        addu   $t0, $a0, $t4
        addu   $a3, $a1, $t4
        move   $v1, $a2
        move   $t2, $zero

loc_24:                                # CODE XREF: m+70
        mtc1  $zero, $f0
        move   $v0, $zero
        mtc1  $zero, $f1
        or     $at, $zero
        swc1  $f0, 4($v1)
        swc1  $f1, 0($v1)

loc_3C:                                # CODE XREF: m+5C
        lwc1  $f4, 4($t0)
        lwc1  $f2, 4($a3)

```

```

lwc1    $f5, 0($t0)
lwc1    $f3, 0($a3)
addiu   $v0, 1
mul.d   $f2, $f4, $f2
add.d   $f0, $f2
swc1    $f0, 4($v1)
bne     $v0, $t1, loc_3C
swc1    $f1, 0($v1)
addiu   $t2, 1
addiu   $v1, 8
addiu   $t0, 8
bne     $t2, $t3, loc_24
addiu   $a3, 8
addiu   $t4, 0x320
bne     $t4, $t5, loc_14
addiu   $a2, 0x320
jr      $ra
or      $at, $zero

```

Responda [G.1.12 on page 1263](#).

18.9.3 Ejercicio #3

Lo que hace el código?

Listing 18.48: Con optimización MSVC 2010

```

_array$ = 8
_x$ = 12
_y$ = 16
_f    PROC
    mov    eax, DWORD PTR _x$[esp-4]
    mov    edx, DWORD PTR _y$[esp-4]
    mov    ecx, eax
    shl    ecx, 4
    sub    ecx, eax
    lea    eax, DWORD PTR [edx+ecx*8]
    mov    ecx, DWORD PTR _array$[esp-4]
    fld    QWORD PTR [ecx+eax*8]
    ret    0
_f    ENDP

```

Listing 18.49: Sin optimización Keil 6/2013 (Modo ARM)

f PROC	
RSB	r1,r1,r1,LSL #4
ADD	r0,r0,r1,LSL #6

ADD	r1,r0,r2,LSL #3
LDM	r1,{r0,r1}
BX	lr
ENDP	

Listing 18.50: Sin optimización Keil 6/2013 (Modo Thumb)

```
f PROC
    MOVS    r3,#0xf
    LSLS    r3,r3,#6
    MULS    r1,r3,r1
    ADDS    r0,r1,r0
    LSLS    r1,r2,#3
    ADDS    r1,r0,r1
    LDM     r1,{r0,r1}
    BX     lr
    ENDP
```

Listing 18.51: Con optimización GCC 4.9 (ARM64)

```
f:
    sxtw   x1, w1
    add    x0, x0, x2, sxtw 3
    lsl    x2, x1, 10
    sub    x1, x2, x1, lsl 6
    ldr    d0, [x0,x1]
    ret
```

Listing 18.52: Con optimización GCC 4.4.5 (MIPS) (IDA)

```
f:
    sll    $v0, $a1, 10
    sll    $a1, 6
    subu   $a1, $v0, $a1
    addu   $a1, $a0, $a1
    sll    $a2, 3
    addu   $a1, $a2
    lwc1   $f0, 4($a1)
    or     $at, $zero
    lwc1   $f1, 0($a1)
    jr     $ra
    or     $at, $zero
```

Responda [G.1.12 on page 1263](#)

18.9.4 Ejercicio #4

Lo que hace el código?

Listing 18.53: Con optimización MSVC 2010

```
_array$ = 8
_x$ = 12
_y$ = 16
_z$ = 20
_f PROC
    mov     eax, DWORD PTR _x$[esp-4]
    mov     edx, DWORD PTR _y$[esp-4]
    mov     ecx, eax
    shl     ecx, 4
    sub     ecx, eax
    lea     eax, DWORD PTR [edx+ecx*4]
    mov     ecx, DWORD PTR _array$[esp-4]
    lea     eax, DWORD PTR [eax+eax*4]
    shl     eax, 4
    add     eax, DWORD PTR _z$[esp-4]
    mov     eax, DWORD PTR [ecx+eax*4]
    ret     0
_f ENDP
```

Listing 18.54: Sin optimización Keil 6/2013 (Modo ARM)

```
f PROC
    RSB      r1,r1,r1,LSL #4
    ADD      r1,r1,r1,LSL #2
    ADD      r0,r0,r1,LSL #8
    ADD      r1,r2,r2,LSL #2
    ADD      r0,r0,r1,LSL #6
    LDR      r0,[r0,r3,LSL #2]
    BX       lr
ENDP
```

Listing 18.55: Sin optimización Keil 6/2013 (Modo Thumb)

```
f PROC
    PUSH    {r4,lr}
    MOVS    r4,#0x4b
    LSLS    r4,r4,#8
    MULS    r1,r4,r1
    ADDS    r0,r1,r0
    MOVS    r1,#0xff
    ADDS    r1,r1,#0x41
    MULS    r2,r1,r2
    ADDS    r0,r0,r2
    LSLS    r1,r3,#2
    LDR     r0,[r0,r1]
    POP     {r4,pc}
```

ENDP

Listing 18.56: Con optimización GCC 4.9 (ARM64)

```
f:
    sxtw    x2, w2
    mov     w4, 19200
    add     x2, x2, x2, lsl 2
    smull   x1, w1, w4
    lsl     x2, x2, 4
    add     x3, x2, x3, sxtw
    add     x0, x0, x3, lsl 2
    ldr     w0, [x0,x1]
    ret
```

Listing 18.57: Con optimización GCC 4.4.5 (MIPS) (IDA)

```
f:
    sll     $v0, $a1, 10
    sll     $a1, 8
    addu   $a1, $v0
    sll     $v0, $a2, 6
    sll     $a2, 4
    addu   $a2, $v0
    sll     $v0, $a1, 4
    subu   $a1, $v0, $a1
    addu   $a2, $a3
    addu   $a1, $a0, $a1
    sll     $a2, 2
    addu   $a1, $a2
    lw      $v0, 0($a1)
    jr     $ra
    or      $at, $zero
```

Responda [G.1.12 on page 1264](#)

18.9.5 Ejercicio #5

Lo que hace el código?

Listing 18.58: Con optimización MSVC 2012 /GS-

```
COMM _tbl:DWORD:064H
tv759 = -4      ; size = 4
_main PROC
    push    ecx
```

```

push    ebx
push    ebp
push    esi
xor    edx, edx
push    edi
xor    esi, esi
xor    edi, edi
xor    ebx, ebx
xor    ebp, ebp
mov     DWORD PTR tv759[esp+20], edx
mov     eax, OFFSET _tbl+4
npad    8 ; align next label
$LL6@main:
lea     ecx, DWORD PTR [edx+edx]
mov     DWORD PTR [eax+4], ecx
mov     ecx, DWORD PTR tv759[esp+20]
add    DWORD PTR tv759[esp+20], 3
mov     DWORD PTR [eax+8], ecx
lea     ecx, DWORD PTR [edx*4]
mov     DWORD PTR [eax+12], ecx
lea     ecx, DWORD PTR [edx*8]
mov     DWORD PTR [eax], edx
mov     DWORD PTR [eax+16], ebp
mov     DWORD PTR [eax+20], ebx
mov     DWORD PTR [eax+24], edi
mov     DWORD PTR [eax+32], esi
mov     DWORD PTR [eax-4], 0
mov     DWORD PTR [eax+28], ecx
add    eax, 40
inc    edx
add    ebp, 5
add    ebx, 6
add    edi, 7
add    esi, 9
cmp    eax, OFFSET _tbl+404
jl    SHORT $LL6@main
pop    edi
pop    esi
pop    ebp
xor    eax, eax
pop    ebx
pop    ecx
ret    0
_main  ENDP

```

Listing 18.59: Sin optimización Keil 6/2013 (Modo ARM)

main PROC

```

LDR      r12, |L0.60|
MOV      r1,#0
|L0.8|
ADD      r2,r1,r1,LSL #2
MOV      r0,#0
ADD      r2,r12,r2,LSL #3
|L0.20|
MUL      r3,r1,r0
STR      r3,[r2,r0,LSL #2]
ADD      r0,r0,#1
CMP      r0,#0xa
BLT      |L0.20|
ADD      r1,r1,#1
CMP      r1,#0xa
MOVGE   r0,#0
BLT      |L0.8|
BX      lr
ENDP

|L0.60|
DCD      ||.bss||

AREA  ||.bss||, DATA, NOINIT, ALIGN=2

tbl
%
400

```

Listing 18.60: Sin optimización Keil 6/2013 (Modo Thumb)

```

main PROC
    PUSH    {r4,r5,lr}
    LDR     r4, |L0.40|
    MOVS   r1,#0
|L0.6|
    MOVS   r2,#0x28
    MULS   r2,r1,r2
    MOVS   r0,#0
    ADDS   r3,r2,r4
|L0.14|
    MOVS   r2,r1
    MULS   r2,r0,r2
    LSLS   r5,r0,#2
    ADDS   r0,r0,#1
    CMP    r0,#0xa
    STR    r2,[r3,r5]
    BLT    |L0.14|
    ADDS   r1,r1,#1
    CMP    r1,#0xa

```

```

BLT      |L0.6|
MOVS    r0,#0
POP     {r4,r5,pc}
ENDP

|L0.40| DCW      0x0000
          DCD      ||.bss||

          AREA ||.bss||, DATA, NOINIT, ALIGN=2

tbl
%        400

```

Listing 18.61: Sin optimización GCC 4.9 (ARM64)

```

.main: .comm  tbl,400,8
       sub    sp, sp, #16
       str   wZR, [sp,12]
       b     .L2
.L5:   str   wZR, [sp,8]
       b     .L3
.L4:   ldr   w1, [sp,12]
       ldr   w0, [sp,8]
       mul   w3, w1, w0
       adrp  x0, tbl
       add   x2, x0, :lo12:tbl
       ldrsw x4, [sp,8]
       ldrsw x1, [sp,12]
       mov   x0, x1
       lsl   x0, x0, 2
       add   x0, x0, x1
       lsl   x0, x0, 1
       add   x0, x0, x4
       str   w3, [x2,x0,lsl 2]
       ldr   w0, [sp,8]
       add   w0, w0, 1
       str   w0, [sp,8]
.L3:   ldr   w0, [sp,8]
       cmp   w0, 9
       ble   .L4
       ldr   w0, [sp,12]
       add   w0, w0, 1
       str   w0, [sp,12]

```

.L2:

```

ldr    w0, [sp,12]
cmp    w0, 9
ble   .L5
mov    w0, 0
add    sp, sp, 16
ret

```

Listing 18.62: Sin optimización GCC 4.4.5 (MIPS) (IDA)

```

main:

var_18      = -0x18
var_10      = -0x10
var_C       = -0xC
var_4       = -4

addiu   $sp, -0x18
sw      $fp, 0x18+var_4($sp)
move   $fp, $sp
la      $gp, __gnu_local_gp
sw      $gp, 0x18+var_18($sp)
sw      $zero, 0x18+var_C($fp)
b       loc_A0
or      $at, $zero

loc_24:          # CODE XREF: main+AC
sw      $zero, 0x18+var_10($fp)
b       loc_7C
or      $at, $zero

loc_30:          # CODE XREF: main+88
lw      $v0, 0x18+var_C($fp)
lw      $a0, 0x18+var_10($fp)
lw      $a1, 0x18+var_C($fp)
lw      $v1, 0x18+var_10($fp)
or      $at, $zero
mult   $a1, $v1
mflo   $a2
lw      $v1, (tbl & 0xFFFF)($gp)
sll    $v0, 1
sll    $a1, $v0, 2
addu   $v0, $a1
addu   $v0, $a0
sll    $v0, 2
addu   $v0, $v1, $v0
sw      $a2, 0($v0)
lw      $v0, 0x18+var_10($fp)

```

```

        or      $at, $zero
        addiu   $v0, 1
        sw      $v0, 0x18+var_10($fp)

loc_7C:                                # CODE XREF: main+28
        lw      $v0, 0x18+var_10($fp)
        or      $at, $zero
        slti   $v0, 0xA
        bnez   $v0, loc_30
        or      $at, $zero
        lw      $v0, 0x18+var_C($fp)
        or      $at, $zero
        addiu   $v0, 1
        sw      $v0, 0x18+var_C($fp)

loc_A0:                                # CODE XREF: main+1C
        lw      $v0, 0x18+var_C($fp)
        or      $at, $zero
        slti   $v0, 0xA
        bnez   $v0, loc_24
        or      $at, $zero
        move   $v0, $zero
        move   $sp, $fp
        lw      $fp, 0x18+var_4($sp)
        addiu   $sp, 0x18
        jr      $ra
        or      $at, $zero

.comm tbl:0x64                         # DATA XREF: main+4C

```

Responda [G.1.12 on page 1264](#)

Capítulo 19

Manipulando bit(s) específicas

19.1

19.1.1 x86

```
HANDLE fh;  
  
fh=CreateFile ("file", GENERIC_WRITE | GENERIC_READ, ↴  
↳ FILE_SHARE_READ, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL ↴  
↳ , NULL);
```

(MSVC 2010):

Listing 19.1: MSVC 2010

```
push    0  
push    128  
↳ 00000080H      ; ↴  
push    4  
push    0  
push    1  
push    -1073741824 ; ↴  
↳ c0000000H  
push    OFFSET $SG78813  
call    DWORD PTR __imp__CreateFileA@28  
mov     DWORD PTR _fh$[ebp], eax
```

WinNT.h:

Listing 19.2: WinNT.h

#define GENERIC_READ	(0x80000000L)
#define GENERIC_WRITE	(0x40000000L)
#define GENERIC_EXECUTE	(0x20000000L)
#define GENERIC_ALL	(0x10000000L)

,GENERIC_READ | GENERIC_WRITE = 0x80000000 | 0x40000000 = 0xC0000000
CreateFile()¹.

Listing 19.3: KERNEL32.DLL (Windows XP SP3 x86)

```
.text:7C83D429          test    byte ptr [ebp+dwDesiredAccess+3], 40h
.text:7C83D42D          mov     [ebp+var_8], 1
.text:7C83D434          jz     short loc_7C83D417
.text:7C83D436          jmp     loc_7C810817
```

(7.3.1 on page 100)).

```
if ((dwDesiredAccess&0x40000000) == 0) goto loc_7C83D417
```

```
#include <stdio.h>
#include <fcntl.h>

void main()
{
    int handle;

    handle=open ("file", O_RDWR | O_CREAT);
}
```

:

Listing 19.4: GCC 4.4.1

```
main          public main
              proc near

var_20        = dword ptr -20h
var_1C        = dword ptr -1Ch
var_4         = dword ptr -4

              push    ebp
              mov     ebp, esp
              and     esp, 0FFFFFFF0h
              sub     esp, 20h
              mov     [esp+20h+var_1C], 42h
```

¹[msdn.microsoft.com/en-us/library/aa363858\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa363858(VS.85).aspx)

```

        mov      [esp+20h+var_20], offset aFile ; "file"
        call     _open
        mov      [esp+20h+var_4], eax
        leave
        retn
main      endp

```

Listing 19.5: open() (libc.so.6)

```

.text:000BE69B          mov      edx, [esp+4+mode] ; mode
.text:000BE69F          mov      ecx, [esp+4+flags] ; ↴
    ↴ flags
.text:000BE6A3          mov      ebx, [esp+4+filename] ; ↴
    ↴ filename
.text:000BE6A7          mov      eax, 5
.text:000BE6AC          int     80h                  ; LINUX ↴
    ↴ - sys_open

```

N.B.

Listing 19.6: do_filp_open() (linux kernel 2.6.31)

```

do_filp_open    proc near
...
        push    ebp
        mov     ebp, esp
        push    edi
        push    esi
        push    ebx
        mov     ebx, ecx
        add     ebx, 1
        sub     esp, 98h
        mov     esi, [ebp+arg_4] ; acc_mode ()
        test   bl, 3
        mov     [ebp+var_80], eax ; dfd ()
        mov     [ebp+var_7C], edx ; pathname ()
        mov     [ebp+var_78], ecx ; open_flag ()
        jnz    short loc_C01EF684
        mov     ebx, ecx         ; ebx <- open_flag

```

:

Listing 19.7: do_filp_open() (linux kernel 2.6.31)

loc_C01EF6B4:	;	CODE XREF: ↴
↴ do_filp_open+4F		
test bl, 40h		; O_CREAT

```

    jnz    loc_C01EF810
    mov    edi, ebx
    shr    edi, 11h
    xor    edi, 1
    and    edi, 1
    test   ebx, 10000h
    jz     short loc_C01EF6D3
    or     edi, 2

```

19.1.2 ARM

Listing 19.8: linux kernel 3.8.0

```

struct file *do_filp_open(int dfd, struct filename *pathname,
                         const struct open_flags *op)
{
...
    filp = path_openat(dfd, pathname, &nd, op, flags | ↴
        ↴ LOOKUP_RCU);
...
}

static struct file *path_openat(int dfd, struct filename *↘
    ↴ pathname,
        struct nameidata *nd, const struct open_flags *↘
    ↴ op, int flags)
{
...
    error = do_last(nd, &path, file, op, &opened, pathname) ↴
    ↴ ;
...
}

static int do_last(struct nameidata *nd, struct path *path,
                   struct file *file, const struct open_flags *↘
    ↴ op,
                   int *opened, struct filename *name)
{
...
    if (!(open_flag & O_CREAT)) {
...
        error = lookup_fast(nd, path, &inode);
...
    } else {
...
        error = complete_walk(nd);
    }
}

```

```

    }
...
}
```

Listing 19.9: do_last() (vmlinux)

```

...
.text:C0169EA8          MOV      R9, R3 ; R3 - ↴
    ↳ (4th argument) open_flag
...
.text:C0169ED4          LDR      R6, [R9] ; R6 - ↴
    ↳ open_flag
...
.text:C0169F68          TST      R6, #0x40 ; ↴
    ↳ jumpable C0169F00 default case
.text:C0169F6C          BNE      loc_C016A128
.text:C0169F70          LDR      R2, [R4,#0x10]
.text:C0169F74          ADD      R12, R4, #8
.text:C0169F78          LDR      R3, [R4,#0xC]
.text:C0169F7C          MOV      R0, R4
.text:C0169F80          STR      R12, [R11,#]
    ↳ var_50]
.text:C0169F84          LDRB     R3, [R2,R3]
.text:C0169F88          MOV      R2, R8
.text:C0169F8C          CMP      R3, #0
.text:C0169F90          ORRNE   R1, R1, #3
.text:C0169F94          STRNE   R1, [R4,#0x24]
.text:C0169F98          ANDS    R3, R6, #0
    ↳ x200000
.text:C0169F9C          MOV      R1, R12
.text:C0169FA0          LDRNE   R3, [R4,#0x24]
.text:C0169FA4          ANDNE   R3, R3, #1
.text:C0169FA8          EORNE   R3, R3, #1
.text:C0169FAC          STR      R3, [R11,#var_54]
    ↳ ]
.text:C0169FB0          SUB     R3, R11, #- ↴
    ↳ var_38
.text:C0169FB4          BL      lookup_fast
...
.text:C016A128 loc_C016A128           ; CODE ↴
    ↳ XREF: do_last.isra.14+DC
.text:C016A128          MOV      R0, R4
.text:C016A12C          BL      complete_walk
...
```

TST

0_CREAT

19.2

:

```
#include <stdio.h>

#define IS_SET(flag, bit)      ((flag) & (bit))
#define SET_BIT(var, bit)      ((var) |= (bit))
#define REMOVE_BIT(var, bit)   ((var) &= ~(bit))

int f(int a)
{
    int rt=a;

    SET_BIT (rt, 0x4000);
    REMOVE_BIT (rt, 0x200);

    return rt;
};

int main()
{
    f(0x12340678);
}
```

19.2.1 x86

Sin optimización MSVC

(MSVC 2010):

Listing 19.10: MSVC 2010

```
_rt$ = -4           ; size = 4
_a$ = 8            ; size = 4
_f  PROC
    push  ebp
    mov   ebp, esp
    push  ecx
    mov   eax, DWORD PTR _a$[ebp]
    mov   DWORD PTR _rt$[ebp], eax
    mov   ecx, DWORD PTR _rt$[ebp]
    or    ecx, 16384          ; 00004000H
    mov   DWORD PTR _rt$[ebp], ecx
    mov   edx, DWORD PTR _rt$[ebp]
    and   edx, -513           ; fffffdffH
    mov   DWORD PTR _rt$[ebp], edx
```

```
    mov    eax, DWORD PTR _rt$[ebp]
    mov    esp, ebp
    pop    ebp
    ret    0
_f    ENDP
```

OllyDbg

OllyDbg. :

0x200 (00000000000000000000000000000000) 0.

0x200 0xFFFFFDFF (1111111111111111111111110111111111)

0x4000 (000000000000000010000000000000) 0.

: 0x12340678 (10010001101000000011001111000). :

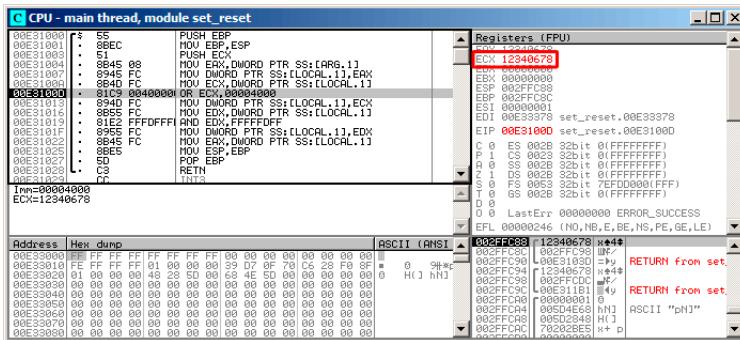


Figura 19.1: OllyDbg: ECX

OR:

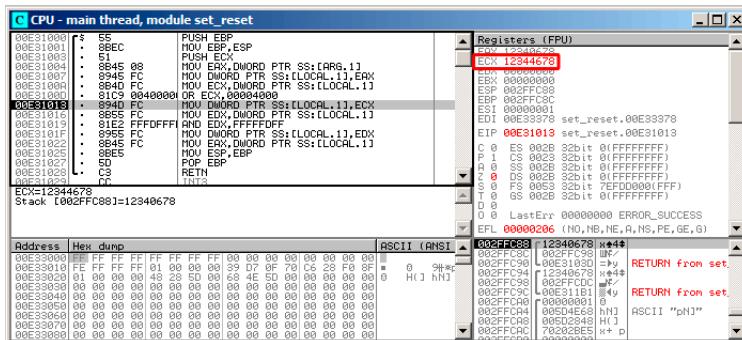


Figura 19.2: OllyDbg: OR

: 0x12344678 (10010001101000100011001111000).

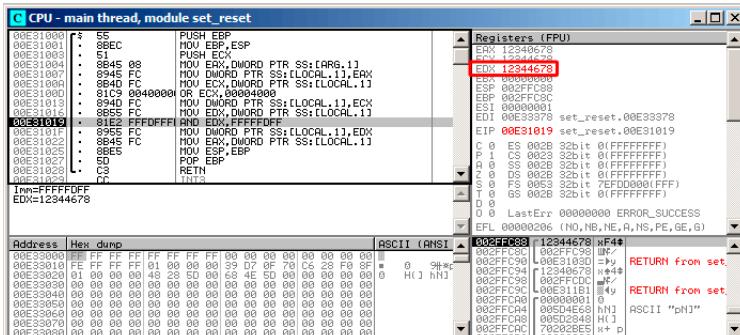


Figura 19.3: OllyDbg: EDX

AND:

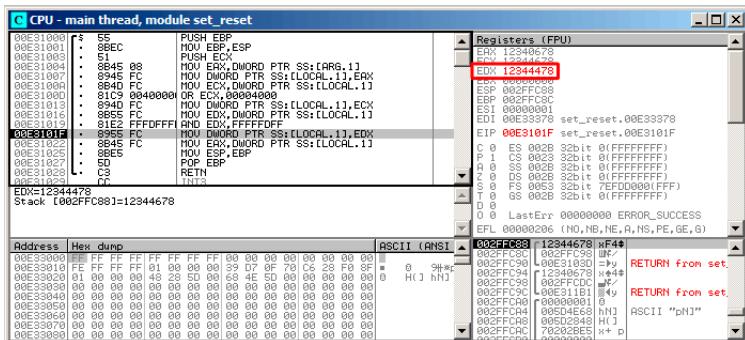


Figura 19.4: OllyDbg: AND

El 10º bit fue limpiado (0, en otras palabras, todos los bits fueron dejados excepto el 10º) y el valor final ahora es 0x12344478 (10010001101000100010001111000).

Con optimización MSVC

Listing 19.11: Con optimización MSVC

```

_a$ = 8          ; size = 4
_f    PROC
    mov     eax, DWORD PTR _a$[esp-4]
    and    eax, -513        ; ffffffdffH
    or     eax, 16384       ; 00004000H
    ret    0
_f    ENDP

```

Sin optimización GCC

Listing 19.12: Sin optimización GCC

```

public f
f
proc near

var_4          = dword ptr -4
arg_0          = dword ptr 8

push    ebp
mov     ebp, esp
sub    esp, 10h

```

```

        mov    eax, [ebp+arg_0]
        mov    [ebp+var_4], eax
        or     [ebp+var_4], 4000h
        and    [ebp+var_4], 0FFFFFDFFh
        mov    eax, [ebp+var_4]
        leave
        retn
f      endp

```

-03:

Con optimización GCC

Listing 19.13: Con optimización GCC

```

public f
proc near

arg_0      = dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+arg_0]
pop    ebp
or     ah, 40h
and    ah, 0FDh
retn
f      endp

```

Spanish text placeholder	
RAX ^{x64}	
	EAX
	AX
AH	AL

N.B.

Con optimización GCC y regparm

Listing 19.14: Con optimización GCC

```

public f
proc near
push    ebp
or     ah, 40h
mov     ebp, esp

```

	and	ah, 0FDh
	pop	ebp
	retn	
f	endp	

19.2.2 ARM + Con optimización Keil 6/2013 (Modo ARM)

Listing 19.15: Con optimización Keil 6/2013 (Modo ARM)

02 0C C0 E3	BIC	R0, R0, #0x200
01 09 80 E3	ORR	R0, R0, #0x4000
1E FF 2F E1	BX	LR

BIC (*Bltwise bit Clear*)

ORR OR en x86.

.

19.2.3 ARM + Con optimización Keil 6/2013 (Modo Thumb)

Listing 19.16: Con optimización Keil 6/2013 (Modo Thumb)

01 21 89 03	MOVS	R1, 0x4000
08 43	ORRS	R0, R1
49 11	ASRS	R1, R1, #5 ; generate 0x200 and ↴ ↓ place to R1
88 43	BICS	R0, R1
70 47	BX	LR

0x200 0x4000, 0x200 .

ASRS (Spanish text placeholder), 0x4000 \gg 5.

19.2.4 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Listing 19.17: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

42 0C C0 E3	BIC	R0, R0, #0x4200
01 09 80 E3	ORR	R0, R0, #0x4000
1E FF 2F E1	BX	LR

REMOVE_BIT (rt, 0x4200);
SET_BIT (rt, 0x4000);

. 0x4200?².

(91 on page 1146).

Con optimización Xcode 4.6.3 (LLVM)

19.2.5 ARM:

:

```
int f(int a)
{
    int rt=a;

    REMOVE_BIT (rt, 0x1234);

    return rt;
};
```

Keil 5.03 :

```
f PROC
    BIC      r0,r0,#0x1000
    BIC      r0,r0,#0x234
    BX       lr
ENDP
```

19.2.6 ARM64: Con optimización GCC (Linaro) 4.9

Con optimización GCC BIC:

Listing 19.18: Con optimización GCC (Linaro) 4.9

```
f:
    and      w0, w0, -513      ; 0xFFFFFFFFFFFFFFFDFF
    orr      w0, w0, 16384     ; 0x4000
    ret
```

19.2.7 ARM64: Sin optimización GCC (Linaro) 4.9

Sin optimización GCC :

Listing 19.19: Sin optimización GCC (Linaro) 4.9

```
f:
```

²LLVM build 2410.2.00 Apple Xcode 4.6.3

```

sub    sp, sp, #32
str    w0, [sp,12]
ldr    w0, [sp,12]
str    w0, [sp,28]
ldr    w0, [sp,28]
orr    w0, w0, 16384    ; 0x4000
str    w0, [sp,28]
ldr    w0, [sp,28]
and    w0, w0, -513     ; 0xFFFFFFFFFFFFFDFF
str    w0, [sp,28]
ldr    w0, [sp,28]
add    sp, sp, 32
ret

```

19.2.8 MIPS

Listing 19.20: Con optimización GCC 4.4.5 (IDA)

```

f:
; $a0=a
            ori      $a0, 0x4000
; $a0=a|0x4000
            li       $v0, 0xFFFFFDFF
            jr       $ra
            and      $v0, $a0, $v0
; : $v0 = $a0&$v0 = a|0x4000 & 0xFFFFFDFF

```

19.3 Shifts

SHL (SHift Left) y SHR (SHift Right) .

[16.1.2 on page 261](#), [16.2.1 on page 267](#).

19.4



(SSpanish text placeholder)

MSB³.

³Most significant bit/byte

```
#include <stdio.h>

float my_abs (float i)
{
    unsigned int tmp=(*(unsigned int*)&i) & 0xFFFFFFFF;
    return *(float*)&tmp;
};

float set_sign (float i)
{
    unsigned int tmp=(*(unsigned int*)&i) | 0x80000000;
    return *(float*)&tmp;
};

float negate (float i)
{
    unsigned int tmp=(*(unsigned int*)&i) ^ 0x80000000;
    return *(float*)&tmp;
};

int main()
{
    printf ("my_abs():\n");
    printf ("%f\n", my_abs (123.456));
    printf ("%f\n", my_abs (-456.123));
    printf ("set_sign():\n");
    printf ("%f\n", set_sign (123.456));
    printf ("%f\n", set_sign (-456.123));
    printf ("negate():\n");
    printf ("%f\n", negate (123.456));
    printf ("%f\n", negate (-456.123));
}
```

19.4.1

0	0	0
0	1	1
1	0	1
1	1	0

19.4.2 x86

:

Listing 19.21: Con optimización MSVC 2012

```

_tmp$ = 8
_i$ = 8
_my_abs PROC
    and    DWORD PTR _i$[esp-4], 2147483647 ; ↴
    ↳ ffffffffH
    fld    DWORD PTR _tmp$[esp-4]
    ret    0
_my_abs ENDP

_tmp$ = 8
_i$ = 8
_set_sign PROC
    or     DWORD PTR _i$[esp-4], -2147483648 ; ↴
    ↳ 80000000H
    fld    DWORD PTR _tmp$[esp-4]
    ret    0
_set_sign ENDP

_tmp$ = 8
_i$ = 8
_negate PROC
    xor    DWORD PTR _i$[esp-4], -2147483648 ; ↴
    ↳ 80000000H
    fld    DWORD PTR _tmp$[esp-4]
    ret    0
_negate ENDP

```

AND y OR. XOR.

:

Listing 19.22: Con optimización MSVC 2012 x64

```

tmp$ = 8
i$ = 8
my_abs PROC
    movss  DWORD PTR [rsp+8], xmm0
    mov    eax, DWORD PTR i$[rsp]
    btr    eax, 31
    mov    DWORD PTR tmp$[rsp], eax
    movss  xmm0, DWORD PTR tmp$[rsp]
    ret    0
my_abs ENDP
_TEXT ENDS

tmp$ = 8
i$ = 8

```

```

set_sign PROC
    movss  DWORD PTR [rsp+8], xmm0
    mov    eax, DWORD PTR i$[rsp]
    bts   eax, 31
    mov    DWORD PTR tmp$[rsp], eax
    movss  xmm0, DWORD PTR tmp$[rsp]
    ret    0
set_sign ENDP

tmp$ = 8
i$ = 8
negate PROC
    movss  DWORD PTR [rsp+8], xmm0
    mov    eax, DWORD PTR i$[rsp]
    btc   eax, 31
    mov    DWORD PTR tmp$[rsp], eax
    movss  xmm0, DWORD PTR tmp$[rsp]
    ret    0
negate ENDP

```

19.4.3 MIPS

GCC 4.4.5 para MIPS :

Listing 19.23: Con optimización GCC 4.4.5 (IDA)

```

my_abs:
; 1:
        mfc1    $v1, $f12
        li      $v0, 0x7FFFFFFF
; $v0=0x7FFFFFFF
; :
        and     $v0, $v1
; 1:
        mtc1    $v0, $f0
;
        jr      $ra
        or      $at, $zero ; branch delay slot

set_sign:
; 1:
        mfc1    $v0, $f12
        lui    $v1, 0x8000
; $v1=0x80000000

```

```

; :
        or      $v0, $v1, $v0
; 1:
        mtc1   $v0, $f0
;
        jr      $ra
        or      $at, $zero ; branch delay slot

negate:
; 1:
        mfc1   $v0, $f12
        lui    $v1, 0x8000
; $v1=0x80000000
; :
        xor    $v0, $v1, $v0
; 1:
        mtc1   $v0, $f0
;
        jr      $ra
        or      $at, $zero ; branch delay slot

```

19.4.4 ARM

Con optimización Keil 6/2013 (Modo ARM)

Listing 19.24: Con optimización Keil 6/2013 (Modo ARM)

```

my_abs PROC
; :
        BIC    r0,r0,#0x80000000
        BX    lr
        ENDP

set_sign PROC
; :
        ORR    r0,r0,#0x80000000
        BX    lr
        ENDP

negate PROC
; :
        EOR    r0,r0,#0x80000000
        BX    lr
        ENDP

```

- («Exclusive OR»).

Con optimización Keil 6/2013 (Modo Thumb)

Listing 19.25: Con optimización Keil 6/2013 (Modo Thumb)

```

my_abs PROC
    LSLS      r0,r0,#1
; r0=i<<1
    LSRS      r0,r0,#1
; r0=(i<<1)>>1
    BX       lr
    ENDP

set_sign PROC
    MOVS     r1,#1
; r1=1
    LSLS     r1,r1,#31
; r1=1<<31=0x80000000
    ORRS     r0,r0,r1
; r0=r0 | 0x80000000
    BX       lr
    ENDP

negate PROC
    MOVS     r1,#1
; r1=1
    LSLS     r1,r1,#31
; r1=1<<31=0x80000000
    EORS     r0,r0,r1
; r0=r0 ^ 0x80000000
    BX       lr
    ENDP

```

: $1 << 31 = 0x80000000$.

: $(i << 1) >> 1 ..$

Con optimización GCC 4.6.3 (Raspberry Pi, Modo ARM)

Listing 19.26: Con optimización GCC 4.6.3 para Raspberry Pi (Modo ARM)

```

my_abs
; :
; :           FMRS     R2, S0
; :           BIC      R3, R2, #0x80000000
; :           FMSR    S0, R3

```

```

        BX      LR
set_sign
; :
; :          FMRS    R2, S0
; :          ORR     R3, R2, #0x80000000
; :          FMSR   S0, R3
; :          BX     LR

negate
; :
; :          FMRS    R2, S0
; :          ADD    R3, R2, #0x80000000
; :          FMSR   S0, R3
; :          BX     LR

```

my_abs() y set_sign() negate()? ?

ADD register, 0x80000000 XOR register, 0x80000000.? :1234567+
10000 = 1244567 (). . .

19.5

«population count»⁴.

```
#include <stdio.h>

#define IS_SET(flag, bit) ((flag) & (bit))

int f(unsigned int a)
{
    int i;
    int rt=0;

    for (i=0; i<32; i++)
        if (IS_SET (a, 1<<i))
            rt++;

    return rt;
}
```

```
int main()
{
    f(0x12345678); // test
};
```

$1 \ll i$ $i = 0 \dots 31$:

C/C++			
$1 \ll 0$	1	1	1
$1 \ll 1$	2^1	2	2
$1 \ll 2$	2^2	4	4
$1 \ll 3$	2^3	8	8
$1 \ll 4$	2^4	16	0x10
$1 \ll 5$	2^5	32	0x20
$1 \ll 6$	2^6	64	0x40
$1 \ll 7$	2^7	128	0x80
$1 \ll 8$	2^8	256	0x100
$1 \ll 9$	2^9	512	0x200
$1 \ll 10$	2^{10}	1024	0x400
$1 \ll 11$	2^{11}	2048	0x800
$1 \ll 12$	2^{12}	4096	0x1000
$1 \ll 13$	2^{13}	8192	0x2000
$1 \ll 14$	2^{14}	16384	0x4000
$1 \ll 15$	2^{15}	32768	0x8000
$1 \ll 16$	2^{16}	65536	0x10000
$1 \ll 17$	2^{17}	131072	0x20000
$1 \ll 18$	2^{18}	262144	0x40000
$1 \ll 19$	2^{19}	524288	0x80000
$1 \ll 20$	2^{20}	1048576	0x100000
$1 \ll 21$	2^{21}	2097152	0x200000
$1 \ll 22$	2^{22}	4194304	0x400000
$1 \ll 23$	2^{23}	8388608	0x800000
$1 \ll 24$	2^{24}	16777216	0x1000000
$1 \ll 25$	2^{25}	33554432	0x2000000
$1 \ll 26$	2^{26}	67108864	0x4000000
$1 \ll 27$	2^{27}	134217728	0x8000000
$1 \ll 28$	2^{28}	268435456	0x10000000
$1 \ll 29$	2^{29}	536870912	0x20000000
$1 \ll 30$	2^{30}	1073741824	0x40000000
$1 \ll 31$	2^{31}	2147483648	0x80000000

ssl_private.h Apache 2.4.6:

```
/**
```

```
* Define the SSL options
*/
#define SSL_OPT_NONE          (0)
#define SSL_OPT_RELSET        (1<<0)
#define SSL_OPT_STDENVVARS    (1<<1)
#define SSL_OPT_EXPORTCERTDATA (1<<3)
#define SSL_OPT_FAKEBASICAUTH (1<<4)
#define SSL_OPT_STRICTREQUIRE (1<<5)
#define SSL_OPT_OPTRENEGOTIATE (1<<6)
#define SSL_OPT_LEGACYDNFORMAT (1<<7)
```

19.5.1 x86

MSVC

(MSVC 2010):

Listing 19.27: MSVC 2010

```
_rt$ = -8           ; size = 4
_i$ = -4           ; size = 4
_a$ = 8            ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 8
    mov     DWORD PTR _rt$[ebp], 0
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN4@f
$LN3@f:
    mov     eax, DWORD PTR _i$[ebp]    ; i
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN4@f:
    cmp     DWORD PTR _i$[ebp], 32      ; 00000020H
    jge     SHORT $LN2@f              ; ?
    mov     edx, 1
    mov     ecx, DWORD PTR _i$[ebp]
    shl     edx, cl                  ; EDX=EDX<<CL
    and     edx, DWORD PTR _a$[ebp]
    je      SHORT $LN1@f              ; ?
    ;
    mov     eax, DWORD PTR _rt$[ebp]
    add     eax, 1                  ; rt
    mov     DWORD PTR _rt$[ebp], eax
```

```
$LN1@f:  
    jmp      SHORT $LN3@f  
$LN2@f:  
    mov      eax, DWORD PTR _rt$[ebp]  
    mov      esp, ebp  
    pop      ebp  
    ret      0  
_f      ENDP
```

OllyDbg

OllyDbg. 0x12345678.

$i = 1, i \text{ ECX}$:

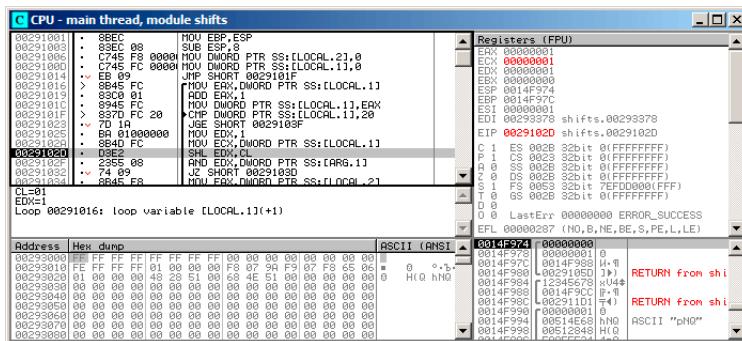
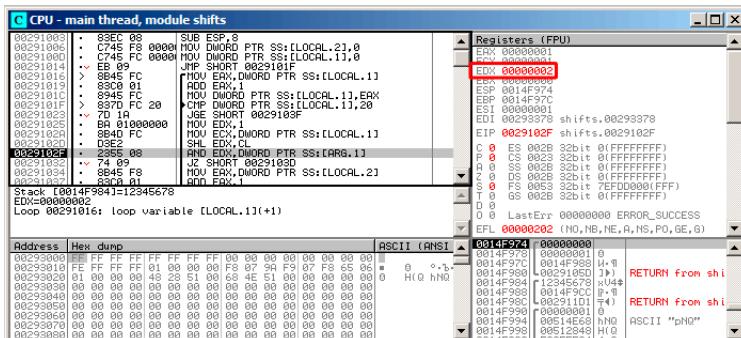


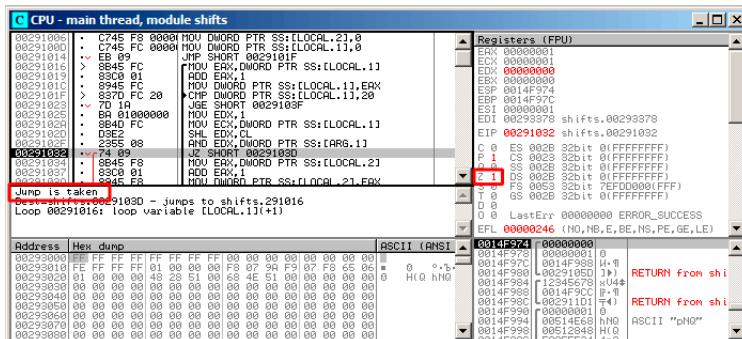
Figura 19.5: OllyDbg: $i = 1, i \text{ ECX}$

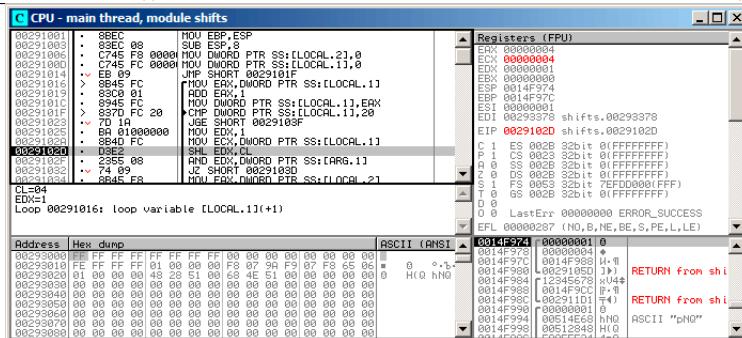
EDX 1. SHL.

SHL:

Figura 19.6: OllyDbg: $i = 1$, $EDX = 1 \ll 1 = 2$ EDX $1 \ll 1$ (o 2)..

AND ZF 1, (0x12345678) 2 0:

Figura 19.7: OllyDbg: $i = 1$, ($ZF = 1$)

Figura 19.8: OllyDbg: $i = 4$, i ECX

EDX =1 << 4 (o 0x10 o 16):

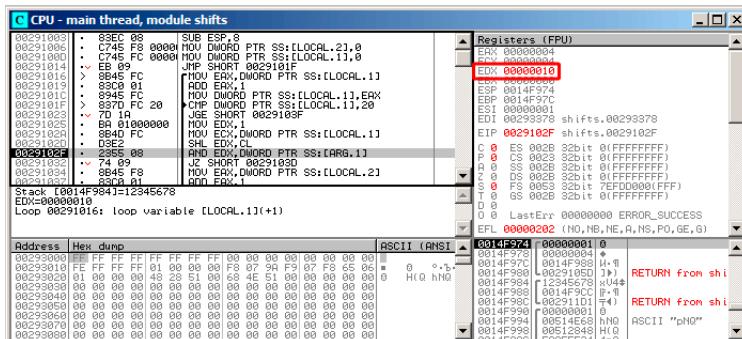
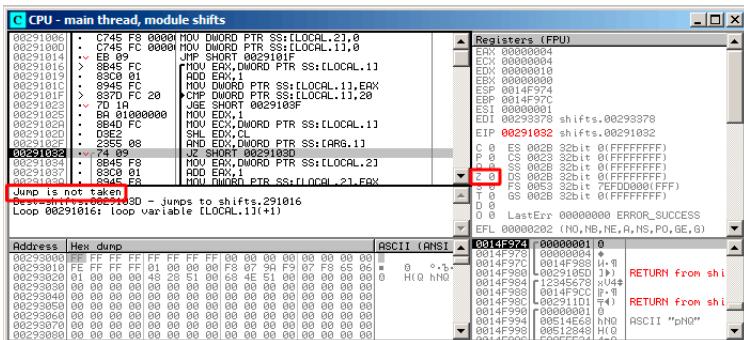


Figura 19.9: OllyDbg: $i = 4$, EDX = $1 \ll 4 = 0x10$

AND:

Figura 19.10: OllyDbg: $i = 4$, ($ZF = 0$) $ZF 0 . , 0x12345678 \& 0x10 = 0x10$.

13. 0x12345678.

GCC

GCC 4.4.1:

Listing 19.28: GCC 4.4.1

```

public f
proc near

rt          = dword ptr -0Ch
i           = dword ptr -8
arg_0       = dword ptr 8

push    ebp
mov     ebp, esp
push    ebx
sub     esp, 10h
mov     [ebp+rt], 0
mov     [ebp+i], 0
jmp     short loc_80483EF

loc_80483D0:
        mov     eax, [ebp+i]
        mov     edx, 1
        mov     ebx, edx
        mov     ecx, eax
        shl     ebx, cl

```

```

        mov    eax, ebx
        and    eax, [ebp+arg_0]
        test   eax, eax
        jz     short loc_80483EB
        add    [ebp+rt], 1
loc_80483EB:
        add    [ebp+i], 1
loc_80483EF:
        cmp    [ebp+i], 1Fh
        jle    short loc_80483D0
        mov    eax, [ebp+rt]
        add    esp, 10h
        pop    ebx
        pop    ebp
        retn
f
        endp

```

19.5.2 x64

:

```

#include <stdio.h>
#include <stdint.h>

#define IS_SET(flag, bit)      ((flag) & (bit))

int f(uint64_t a)
{
    uint64_t i;
    int rt=0;

    for (i=0; i<64; i++)
        if (IS_SET (a, 1ULL<<i))
            rt++;

    return rt;
}

```

Sin optimización GCC 4.8.2

.

Listing 19.29: Sin optimización GCC 4.8.2

```
f:
```

```

    push    rbp
    mov     rbp, rsp
    mov     QWORD PTR [rbp-24], rdi ; a
    mov     DWORD PTR [rbp-12], 0    ; rt=0
    mov     QWORD PTR [rbp-8], 0     ; i=0
    jmp     .L2

.L4:
    mov     rax, QWORD PTR [rbp-8]
    mov     rdx, QWORD PTR [rbp-24]
; RAX = i, RDX = a
    mov     ecx, eax
; ECX = i
    shr     rdx, cl
; RDX = RDX>>CL = a>>i
    mov     rax, rdx
; RAX = RDX = a>>i
    and     eax, 1
; EAX = EAX&1 = (a>>i)&1
    test    rax, rax
; ?
; .
    je     .L3
    add    DWORD PTR [rbp-12], 1    ; rt++
.L3:
    add    QWORD PTR [rbp-8], 1     ; i++
.L2:
    cmp    QWORD PTR [rbp-8], 63    ; i<63?
    jbe    .L4
    mov    eax, DWORD PTR [rbp-12] ; rt
    pop    rbp
    ret

```

Con optimización GCC 4.8.2

Listing 19.30: Con optimización GCC 4.8.2

```

1 f:
2     xor    eax, eax      ;
3     xor    ecx, ecx      ;
4 .L3:
5     mov    rsi, rdi      ;
6     lea    edx, [rax+1]   ; EDX=EAX+1
7 ;
8     shr    rsi, cl       ; RSI=RSI>>CL
9     and    esi, 1        ; ESI=ESI&1
10 ;
11 ;

```

```

12         cmovne  eax, edx
13         add     rcx, 1          ; RCX++
14         cmp     rcx, 64
15         jne     .L3
16         rep    ret           ; AKA fatret

```

. CMOVNE⁵ (CMOVNZ⁶) «rt» EDX («») EAX (« rt »).

: [33.1 on page 589.](#)

REP RET (F3 C3) FATRET MSVC. : [AMD13b, p.15]⁷.

Con optimización MSVC 2010

Listing 19.31: MSVC 2010

```

a$ = 8
f      PROC
; RCX =
        xor     eax, eax
        mov     edx, 1
        lea     r8d, QWORD PTR [rax+64]
; R8D=64
        npad   5
$LL4@f:
        test   rdx, rcx
; ?
; .
        je     SHORT $LN3@f
        inc    eax       ; rt++
$LN3@f:
        rol    rdx, 1   ; RDX=RDX<<1
        dec    r8        ; R8--
        jne   SHORT $LL4@f
        fatret 0
f      ENDP

```

ROL SHL, «rotate left» «shift left», SHL.

: [A.6.3 on page 1241.](#)

R8 ..

:

⁵Conditional MOVe if Not Equal

⁶Conditional MOVe if Not Zero

⁷: <http://go.yurichev.com/17328>

RDX	R8
0x000000000000000000000001	64
0x000000000000000000000002	63
0x000000000000000000000004	62
0x000000000000000000000008	61
...	...
0x400000000000000000000000	2
0x800000000000000000000000	1

FATRET: [19.5.2 on the preceding page.](#)

Con optimización MSVC 2012

Listing 19.32: MSVC 2012

```
a$ = 8
f      PROC
; RCX =
    xor    eax, eax
    mov    edx, 1
    lea    r8d, QWORD PTR [rax+32]
; EDX = 1, R8D = 32
    npad   5
$LL4@f:
;
    test   rdx, rcx
    je     SHORT $LN3@f
    inc    eax      ; rt++
$LN3@f:
    rol    rdx, 1  ; RDX=RDX<<1
; -----
;
    test   rdx, rcx
    je     SHORT $LN11@f
    inc    eax      ; rt++
$LN11@f:
    rol    rdx, 1  ; RDX=RDX<<1
; -----
    dec    r8      ; R8--
    jne    SHORT $LL4@f
    fatret 0
f      ENDP
```

Con optimización MSVC 2012 MSVC 2010, .

19.5.3 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

Listing 19.33: Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```

        MOV      R1, R0
        MOV      R0, #0
        MOV      R2, #1
        MOV      R3, R0
loc_2E54
        TST      R1, R2,LSL R3 ; R1 & (R2<<R3)
        ADD      R3, R3, #1      ; R3++
        ADDNE   R0, R0, #1      ; R0++
        CMP      R3, #32
        BNE      loc_2E54
        BX       LR

```

TST TEST en x86.

(41.2.1 on page 633), LSL (*Logical Shift Left*), LSR (*Logical Shift Right*), ASR (*Arithmetic Shift Right*), ROR (*Rotate Right*) y RRX (*Rotate Right with Extend*) MOV, TST, CMP, ADD, SUB, RSB⁸.

«TST R1 , R2 ,LSL R3» $R1 \wedge (R2 \ll R3)$.

19.5.4 ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

, LSL.W/TST TST, LSL TST.

```

        MOV      R1, R0
        MOVS    R0, #0
        MOV.W   R9, #1
        MOVS    R3, #0
loc_2F7A
        LSL.W   R2, R9, R3
        TST     R2, R1
        ADD.W   R3, R3, #1
        IT NE
        ADDNE  R0, #1
        CMP     R3, #32
        BNE     loc_2F7A
        BX      LR

```

⁸Estas instrucciones también se llaman «data processing instructions»

19.5.5 ARM64 + Con optimización GCC 4.9

[: 19.5.2 on page 419.](#)

Listing 19.34: Con optimización GCC (Linaro) 4.8

```
f:
    mov    w2, 0           ; rt=0
    mov    x5, 1
    mov    w1, w2

.L2:
    lsl    x4, x5, x1      ; w4 = w5<<w1 = 1<<i
    add    w3, w2, 1        ; new_rt=rt+1
    tst    x4, x0          ; (1<<i) & a
    add    w1, w1, 1        ; i++
; ?
; w2=w3  rt=new_rt.
; : w2=w2  rt=rt ()
    csel   w2, w3, w2, ne
    cmp    w1, 64          ; i<64?
    bne   .L2
    mov    w0, w2          ; rt
    ret
```

[: 19.30 on page 420.](#)

CSEL «Conditional SELect». TST W2.

19.5.6 ARM64 + Sin optimización GCC 4.9

[: 19.5.2 on page 419.](#)

Listing 19.35: Sin optimización GCC (Linaro) 4.8

```
f:
    sub    sp, sp, #32
    str    x0, [sp,8]       ; Register Save Area
    str    wzr, [sp,24]     ; rt=0
    str    wzr, [sp,28]     ; i=0
    b     .L2

.L4:
    ldr    w0, [sp,28]
    mov    x1, 1
    lsl    x0, x1, x0      ; X0 = X1<<X0 = 1<<i
    mov    x1, x0
```

```

; X1 = 1<<1
        ldr      x0, [sp,8]
; X0 = a
        and      x0, x1, x0
; X0 = X1&X0 = (1<<i) & a
; X0 ?
        cmp      x0, xzr
        beq     .L3
; rt++
        ldr      w0, [sp,24]
        add      w0, w0, 1
        str      w0, [sp,24]
.L3:
; i++
        ldr      w0, [sp,28]
        add      w0, w0, 1
        str      w0, [sp,28]
.L2:
; i<=63? .L4
        ldr      w0, [sp,28]
        cmp      w0, 63
        ble     .L4
; rt
        ldr      w0, [sp,24]
        add      sp, sp, 32
        ret

```

19.5.7 MIPS

Sin optimización GCC

Listing 19.36: Sin optimización GCC 4.4.5 (IDA)

```

f:
; :
rt          = -0x10
i           = -0xC
var_4       = -4
a           = 0

        addiu   $sp, -0x18
        sw      $fp, 0x18+var_4($sp)
        move    $fp, $sp
        sw      $a0, 0x18+a($fp)
; :
        sw      $zero, 0x18+rt($fp)

```

```

        sw      $zero, 0x18+i($fp)
; :
        b       loc_68
        or     $at, $zero ; branch delay slot, NOP

loc_20:
        li      $v1, 1
        lw      $v0, 0x18+i($fp)
        or     $at, $zero ; load delay slot, NOP
        sllv   $v0, $v1, $v0
; $v0 = 1<<i
        move   $v1, $v0
        lw      $v0, 0x18+a($fp)
        or     $at, $zero ; load delay slot, NOP
        and   $v0, $v1, $v0
; $v0 = a&(1<<i)
; a&(1<<i) ? :
        beqz   $v0, loc_58
        or     $at, $zero
; a&(1<<i)!=0, :
        lw      $v0, 0x18+rt($fp)
        or     $at, $zero ; load delay slot, NOP
        addiu  $v0, 1
        sw      $v0, 0x18+rt($fp)

loc_58:
; i:
        lw      $v0, 0x18+i($fp)
        or     $at, $zero ; load delay slot, NOP
        addiu  $v0, 1
        sw      $v0, 0x18+i($fp)

loc_68:
; 0x20 (32).
; 0x20 (32):
        lw      $v0, 0x18+i($fp)
        or     $at, $zero ; load delay slot, NOP
        slti   $v0, 0x20 # ''
        bnez   $v0, loc_20
        or     $at, $zero ; branch delay slot, NOP
; . rt:
        lw      $v0, 0x18+rt($fp)
        move   $sp, $fp ; load delay slot
        lw      $fp, 0x18+var_4($sp)
        addiu  $sp, 0x18 ; load delay slot
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP

```

Con optimización GCC

. ? : [33.1 on page 589.](#)

Listing 19.37: Con optimización GCC 4.4.5 (IDA)

```

f:
; $a0=a
; $v0:
        move    $v0, $zero
; $v1:
        move    $v1, $zero
        li      $t0, 1
        li      $a3, 32
        sllv   $a1, $t0, $v1
; $a1 = $t0<<$v1 = 1<<i

loc_14:
        and     $a1, $a0
; $a1 = a&(1<<i)
; i:
        addiu  $v1, 1
; loc\_28 a&(1<<i)==0    rt:
        beqz   $a1, loc_28
        addiu  $a2, $v0, 1
; $v0:
        move    $v0, $a2

loc_28:
; i!=32,  loc_14 :
        bne    $v1, $a3, loc_14
        sllv   $a1, $t0, $v1
;
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP

```

19.6 Conclusión**19.6.1**

Listing 19.38: C/C++

```

if (input&0x40)
...

```

Listing 19.39: x86

```
TEST REG, 40h
JNZ is_set
;
```

Listing 19.40: x86

```
TEST REG, 40h
JZ is_cleared
;
```

Listing 19.41: ARM (Modo ARM)

```
TST REG, #0x40
BNE is_set
;
```

19.6.2

Listing 19.42: C/C++

```
if ((value>>n)&1)
    ....
```

Listing 19.43: x86

```
; REG=input_value
; CL=n
SHR REG, CL
AND REG, 1
```

Listing 19.44: C/C++

```
if (value & (1<<n))
    ....
```

Listing 19.45: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
AND input_value, REG
```

19.6.3

Listing 19.46: C/C++

```
value=value|0x40;
```

Listing 19.47: x86

```
OR REG, 40h
```

Listing 19.48: ARM (Modo ARM) y ARM64

```
ORR R0, R0, #0x40
```

19.6.4

Listing 19.49: C/C++

```
value=value|(1<<n);
```

Listing 19.50: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
OR input_value, REG
```

19.6.5

Listing 19.51: C/C++

```
value=value&(~0x40);
```

Listing 19.52: x86

```
AND REG, 0FFFFFFFBFh
```

Listing 19.53: x64

```
AND REG, 0xFFFFFFFFFFFFFFBFh
```

Listing 19.54: ARM (Modo ARM)

```
BIC R0, R0, #0x40
```

19.6.6

Listing 19.55: C/C++

```
value=value&(~(1<<n));
```

Listing 19.56: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
NOT REG
AND input_value, REG
```

19.7 Ejercicios

19.7.1 Ejercicio #1

Lo que hace el código?

Listing 19.57: Con optimización MSVC 2010

```
_a$ = 8
_f PROC
    mov     ecx, DWORD PTR _a$[esp-4]
    mov     eax, ecx
    mov     edx, ecx
    shl     edx, 16          ; 00000010H
    and     eax, 65280        ; 0000ff00H
    or      eax, edx
    mov     edx, ecx
    and     edx, 16711680     ; 00ff0000H
    shr     ecx, 16          ; 00000010H
    or      edx, ecx
    shl     eax, 8
    shr     edx, 8
    or      eax, edx
    ret     0
_f ENDP
```

Listing 19.58: Con optimización Keil 6/2013 (Modo ARM)

```
f PROC
    MOV     r1,#0xff0000
    AND     r1,r1,r0,LSL #8
    MOV     r2,#0xff00
    ORR     r1,r1,r0,LSR #24
```

```

AND      r2,r2,r0,LSR #8
ORR      r1,r1,r2
ORR      r0,r1,r0,LSL #24
BX       lr
ENDP

```

Listing 19.59: Con optimización Keil 6/2013 (Modo Thumb)

```

f PROC
    MOVS    r3,#0xff
    LSLS    r2,r0,#8
    LSLS    r3,r3,#16
    ANDS   r2,r2,r3
    LSRS    r1,r0,#24
    ORRS   r1,r1,r2
    LSRS    r2,r0,#8
    ASRS    r3,r3,#8
    ANDS   r2,r2,r3
    ORRS   r1,r1,r2
    LSLS    r0,r0,#24
    ORRS   r0,r0,r1
    BX     lr
    ENDP

```

Listing 19.60: Con optimización GCC 4.9 (ARM64)

```

f:
    rev    w0, w0
    ret

```

Listing 19.61: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:
    srl    $v0, $a0, 24
    sll    $v1, $a0, 24
    sll    $a1, $a0, 8
    or     $v1, $v0
    lui    $v0, 0xFF
    and   $v0, $a1, $v0
    srl    $a0, 8
    or     $v0, $v1, $v0
    andi  $a0, 0xFF00
    jr    $ra
    or     $v0, $a0

```

Responda: [G.1.13 on page 1264](#).

19.7.2 Ejercicio #2

Lo que hace el código?

Listing 19.62: Con optimización MSVC 2010

```
_a$ = 8 ; size 2
    ↵ = 4
_f PROC
    push    esi
    mov     esi, DWORD PTR _a$[esp]
    xor     ecx, ecx
    push    edi
    lea     edx, DWORD PTR [ecx+1]
    xor     eax, eax
    npad   3 ; align next label
$LL3@f:
    mov     edi, esi
    shr     edi, cl
    add     ecx, 4
    and    edi, 15
    imul   edi, edx
    lea     edx, DWORD PTR [edx+edx*4]
    add     eax, edi
    add     edx, edx
    cmp     ecx, 28
    jle     SHORT $LL3@f
    pop     edi
    pop     esi
    ret    0
_f ENDP
```

Listing 19.63: Con optimización Keil 6/2013 (Modo ARM)

```
f PROC
    MOV    r3,r0
    MOV    r1,#0
    MOV    r2,#1
    MOV    r0,r1
|L0.16|
    LSR    r12,r3,r1
    AND    r12,r12,#0xf
    MLA    r0,r12,r2,r0
    ADD    r1,r1,#4
    ADD    r2,r2,r2,LSL #2
    CMP    r1,#0x1c
    LSL    r2,r2,#1
    BLE    |L0.16|
    BX     lr
```

ENDP

Listing 19.64: Con optimización Keil 6/2013 (Modo Thumb)

```
f PROC
    PUSH    {r4,lr}
    MOVS    r3,r0
    MOVS    r1,#0
    MOVS    r2,#1
    MOVS    r0,r1
|L0.10|
    MOVS    r4,r3
    LSRS    r4,r4,r1
    LSLS    r4,r4,#28
    LSRS    r4,r4,#28
    MULS    r4,r2,r4
    ADDS    r0,r4,r0
    MOVS    r4,#0xa
    MULS    r2,r4,r2
    ADDS    r1,r1,#4
    CMP     r1,#0x1c
    BLE    |L0.10|
    POP    {r4,pc}
ENDP
```

Listing 19.65: Sin optimización GCC 4.9 (ARM64)

```
f:
    sub    sp, sp, #32
    str    w0, [sp,12]
    str    wzr, [sp,28]
    mov    w0, 1
    str    w0, [sp,24]
    str    wzr, [sp,20]
    b     .L2
.L3:
    ldr    w0, [sp,28]
    ldr    w1, [sp,12]
    lsr    w0, w1, w0
    and   w1, w0, 15
    ldr    w0, [sp,24]
    mul   w0, w1, w0
    ldr    w1, [sp,20]
    add   w0, w1, w0
    str    w0, [sp,20]
    ldr    w0, [sp,28]
    add   w0, w0, 4
    str    w0, [sp,28]
```

```

    ldr    w1, [sp,24]
    mov    w0, w1
    lsl    w0, w0, 2
    add    w0, w0, w1
    lsl    w0, w0, 1
    str    w0, [sp,24]
.L2:
    ldr    w0, [sp,28]
    cmp    w0, 28
    ble    .L3
    ldr    w0, [sp,20]
    add    sp, sp, 32
    ret

```

Listing 19.66: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:
    srl    $v0, $a0, 8
    srl    $a3, $a0, 20
    andi   $a3, 0xF
    andi   $v0, 0xF
    srl    $a1, $a0, 12
    srl    $a2, $a0, 16
    andi   $a1, 0xF
    andi   $a2, 0xF
    sll    $t2, $v0, 4
    sll    $v1, $a3, 2
    sll    $t0, $v0, 2
    srl    $t1, $a0, 4
    sll    $t5, $a3, 7
    addu   $t0, $t2
    subu   $t5, $v1
    andi   $t1, 0xF
    srl    $v1, $a0, 28
    sll    $t4, $a1, 7
    sll    $t2, $a2, 2
    sll    $t3, $a1, 2
    sll    $t7, $a2, 7
    srl    $v0, $a0, 24
    addu   $a3, $t5, $a3
    subu   $t3, $t4, $t3
    subu   $t7, $t2
    andi   $v0, 0xF
    sll    $t5, $t1, 3
    sll    $t6, $v1, 8
    sll    $t2, $t0, 2
    sll    $t4, $t1, 1
    sll    $t1, $v1, 3

```

```

addu    $a2, $t7, $a2
subu    $t1, $t6, $t1
addu    $t2, $t0, $t2
addu    $t4, $t5
addu    $a1, $t3, $a1
sll     $t5, $a3, 2
sll     $t3, $v0, 8
sll     $t0, $v0, 3
addu    $a3, $t5
subu    $t0, $t3, $t0
addu    $t4, $t2, $t4
sll     $t3, $a2, 2
sll     $t2, $t1, 6
sll     $a1, 3
addu    $a1, $t4, $a1
subu    $t1, $t2, $t1
addu    $a2, $t3
sll     $t2, $t0, 6
sll     $t3, $a3, 2
andi   $a0, 0xF
addu    $v1, $t1, $v1
addu    $a0, $a1
addu    $a3, $t3
subu    $t0, $t2, $t0
sll     $a2, 4
addu    $a2, $a0, $a2
addu    $v0, $t0, $v0
sll     $a1, $v1, 2
sll     $a3, 5
addu    $a3, $a2, $a3
addu    $v1, $a1
sll     $v0, 6
addu    $v0, $a3, $v0
sll     $v1, 7
jr    $ra
addu    $v0, $v1, $v0

```

Responda: [G.1.13 on page 1264](#).

19.7.3 Ejercicio #3

Listing 19.67: Con optimización MSVC 2010

```

_main PROC
push 278595      ; 00044043H
push OFFSET $SG79792 ; 'caption'
push OFFSET $SG79793 ; 'hello, world!'

```

```

push    0
call    DWORD PTR __imp__MessageBoxA@16
xor     eax, eax
ret    0
_main  ENDP

```

Responda: [G.1.13 on page 1265](#).

19.7.4 Ejercicio #4

Lo que hace el código?

Listing 19.68: Con optimización MSVC 2010

```

_m$ = 8          ; size = 4
_n$ = 12         ; size = 4
_f      PROC
    mov     ecx, DWORD PTR _n$[esp-4]
    xor     eax, eax
    xor     edx, edx
    test   ecx, ecx
    je     SHORT $LN2@f
    push   esi
    mov     esi, DWORD PTR _m$[esp]

$LL3@f:
    test   cl, 1
    je     SHORT $LN1@f
    add    eax, esi
    adc    edx, 0

$LN1@f:
    add    esi, esi
    shr    ecx, 1
    jne    SHORT $LL3@f
    pop    esi

$LN2@f:
    ret    0
_f      ENDP

```

Listing 19.69: Con optimización Keil 6/2013 (Modo ARM)

```

f      PROC
    PUSH   {r4,lr}
    MOV    r3,r0
    MOV    r0,#0
    MOV    r2,r0
    MOV    r12,r0
    B     |L0.48|

```

```
|L0.24|
    TST      r1,#1
    BEQ      |L0.40|
    ADDS    r0,r0,r3
    ADC     r2,r2,r12
|L0.40|
    LSL      r3,r3,#1
    LSR      r1,r1,#1
|L0.48|
    CMP      r1,#0
    MOVEQ   r1,r2
    BNE      |L0.24|
    POP     {r4,pc}
    ENDP
```

Listing 19.70: Con optimización Keil 6/2013 (Modo Thumb)

```
f PROC
    PUSH    {r4,r5,lr}
    MOVS    r3,r0
    MOVS    r0,#0
    MOVS    r2,r0
    MOVS    r4,r0
    B       |L0.24|
|L0.12|
    LSLS    r5,r1,#31
    BEQ     |L0.20|
    ADDS    r0,r0,r3
    ADCS    r2,r2,r4
|L0.20|
    LSLS    r3,r3,#1
    LSRS    r1,r1,#1
|L0.24|
    CMP     r1,#0
    BNE     |L0.12|
    MOVS    r1,r2
    POP     {r4,r5,pc}
    ENDP
```

Listing 19.71: Con optimización GCC 4.9 (ARM64)

```
f:
    mov     w2, w0
    mov     x0, 0
    cbz     w1, .L2
.L3:
    and     w3, w1, 1
    lsr     w1, w1, 1
```

```

    cmp      w3, wzr
    add      x3, x0, x2, uxtw
    lsl      w2, w2, 1
    csel    x0, x3, x0, ne
    cbnz   w1, .L3
.L2:
    ret

```

Listing 19.72: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

mult:
    beqz   $a1, loc_40
    move   $a3, $zero
    move   $a2, $zero
    addu   $t0, $a3, $a0

loc_10:                      # CODE XREF: mult+38
    sltu   $t1, $t0, $a3
    move   $v1, $t0
    andi   $t0, $a1, 1
    addu   $t1, $a2
    beqz   $t0, loc_30
    srl    $a1, 1
    move   $a3, $v1
    move   $a2, $t1

loc_30:                      # CODE XREF: mult+20
    beqz   $a1, loc_44
    sll    $a0, 1
    b     loc_10
    addu   $t0, $a3, $a0
loc_40:                      # CODE XREF: mult
    move   $a2, $zero

loc_44:                      # CODE XREF: mult: ↴
    ↴ loc_30
    move   $v0, $a2
    jr    $ra
    move   $v1, $a3

```

Responda: [G.1.13 on page 1265](#).

Capítulo 20

```
#include <stdint.h>

//  
#define RNG_a 1664525  
#define RNG_c 1013904223

static uint32_t rand_state;

void my_srand (uint32_t init)  
{  
    rand_state=init;  
}

int my_rand ()  
{  
    rand_state=rand_state*RNG_a;  
    rand_state=rand_state+RNG_c;  
    return rand_state & 0x7fff;  
}
```

. [Pre+07].

20.1 x86

Listing 20.1: Con optimización MSVC 2013

```
_BSS      SEGMENT  
_rand_state DD  01H DUP (?)  
_BSS      ENDS  
  
_init$ = 8
```

```

_srand PROC
    mov     eax, DWORD PTR _init$[esp-4]
    mov     DWORD PTR _rand_state, eax
    ret     0
_srand ENDP

_TEXT SEGMENT
_rand PROC
    imul   eax, DWORD PTR _rand_state, 1664525
    add    eax, 1013904223           ; 3c6ef35fH
    mov    DWORD PTR _rand_state, eax
    and    eax, 32767              ; 00007fffH
    ret    0
_rand ENDP

_TEXT ENDS

```

:

Listing 20.2: Sin optimización MSVC 2013

```

_BSS SEGMENT
_rand_state DD 01H DUP (?)
_BSS ENDS

_init$ = 8
_srand PROC
    push   ebp
    mov    ebp, esp
    mov    eax, DWORD PTR _init$[ebp]
    mov    DWORD PTR _rand_state, eax
    pop    ebp
    ret    0
_srand ENDP

_TEXT SEGMENT
_rand PROC
    push   ebp
    mov    ebp, esp
    imul   eax, DWORD PTR _rand_state, 1664525
    mov    DWORD PTR _rand_state, eax
    mov    ecx, DWORD PTR _rand_state
    add    ecx, 1013904223          ; 3c6ef35fH
    mov    DWORD PTR _rand_state, ecx
    mov    eax, DWORD PTR _rand_state
    and    eax, 32767              ; 00007fffH
    pop    ebp
    ret    0

```

```
_rand    ENDP
_TEXT    ENDS
```

20.2 x64

Listing 20.3: Con optimización MSVC 2013 x64

```
_BSS    SEGMENT
rand_state DD 01H DUP (?)
_BSS    ENDS

init$ = 8
my_srand PROC
; ECX =
    mov     DWORD PTR rand_state, ecx
    ret    0
my_srand ENDP

_TEXT    SEGMENT
my_rand PROC
    imul    eax, DWORD PTR rand_state, 1664525      ; ↴
    ↳ 0019660dH
    add     eax, 1013904223                      ; ↴ 3 ↴
    ↳ c6ef35fH
    mov     DWORD PTR rand_state, eax
    and     eax, 32767                           ; 00007 ↴
    ↳ ffff
    ret    0
my_rand ENDP

_TEXT    ENDS
```

20.3 32-bit ARM

Listing 20.4: Con optimización Keil 6/2013 (Modo ARM)

```
my_srand PROC
    LDR    r1, |L0.52|   ; rand_state
    STR    r0,[r1,#0]   ; rand_state
```

```

        BX      lr
        ENDP

my_rand PROC
    LDR    r0, |L0.52| ; rand_state
    LDR    r2, |L0.56| ; RNG_a
    LDR    r1,[r0,#0] ; rand_state
    MUL    r1,r2,r1
    LDR    r2,|L0.60| ; RNG_c
    ADD    r1,r1,r2
    STR    r1,[r0,#0] ; rand_state
; AND 0x7FFF:
    LSL    r0,r1,#17
    LSR    r0,r0,#17
    BX     lr
    ENDP

|L0.52|
    DCD    ||.data||
|L0.56|
    DCD    0x0019660d
|L0.60|
    DCD    0x3c6ef35f

    AREA ||.data||, DATA, ALIGN=2

rand_state
    DCD    0x00000000

```

Parece una operación inútil, pero lo que hace es limpiar los 17 bits altos, dejando intactos los 15 bits bajos, y ese es nuestro objetivo.

Con optimización Keil .

20.4 MIPS

Listing 20.5: Con optimización GCC 4.4.5 (IDA)

```

my_srand:
; rand_state:
        lui     $v0, (rand_state >> 16)
        jr     $ra
        sw     $a0, rand_state
my_rand:
;

```

```

lui      $v1, (rand_state >> 16)
lw       $v0, rand_state
or       $at, $zero ; load delay slot
; 1664525 (RNG_a):
sll      $a1, $v0, 2
sll      $a0, $v0, 4
addu    $a0, $a1, $a0
sll      $a1, $a0, 6
subu    $a0, $a1, $a0
addu    $a0, $v0
sll      $a1, $a0, 5
addu    $a0, $a1
sll      $a0, 3
addu    $v0, $a0, $v0
sll      $a0, $v0, 2
addu    $v0, $a0
; 1013904223 (RNG_c)
;
li       $a0, 0x3C6EF35F
addu    $v0, $a0
; rand_state:
sw       $v0, (rand_state & 0xFFFF)($v1)
jr       $ra
andi    $v0, 0x7FFF ; branch delay slot

```

(0x3C6EF35F o 1013904223). (1664525)?

:

```
#define RNG_a 1664525

int f (int a)
{
    return a*RNG_a;
}
```

Listing 20.6: Con optimización GCC 4.4.5 (IDA)

```
f:
sll      $v1, $a0, 2
sll      $v0, $a0, 4
addu    $v0, $v1, $v0
sll      $v1, $v0, 6
subu    $v0, $v1, $v0
addu    $v0, $a0
sll      $v1, $v0, 5
addu    $v0, $v1
sll      $v0, 3
```

addu	\$a0, \$v0, \$a0
sll	\$v0, \$a0, 2
jr	\$ra
addu	\$v0, \$a0, \$v0 ; branch delay slot

!

20.4.1

Listing 20.7: Con optimización GCC 4.4.5 (objdump)

```
# objdump -D rand_03.o
```

```
...
```

```
00000000 <my_srand>:
```

0:	3c020000	lui	v0,0x0
4:	03e00008	jr	ra
8:	ac440000	sw	a0,0(v0)

```
0000000c <my_rand>:
```

c:	3c030000	lui	v1,0x0
10:	8c620000	lw	v0,0(v1)
14:	00200825	move	at,at
18:	00022880	sll	a1,v0,0x2
1c:	00022100	sll	a0,v0,0x4
20:	00a42021	addu	a0,a1,a0
24:	00042980	sll	a1,a0,0x6
28:	00a42023	subu	a0,a1,a0
2c:	00822021	addu	a0,a0,v0
30:	00042940	sll	a1,a0,0x5
34:	00852021	addu	a0,a0,a1
38:	000420c0	sll	a0,a0,0x3
3c:	00821021	addu	v0,a0,v0
40:	00022080	sll	a0,v0,0x2
44:	00441021	addu	v0,v0,a0
48:	3c043c6e	lui	a0,0x3c6e
4c:	3484f35f	ori	a0,a0,0xf35f
50:	00441021	addu	v0,v0,a0
54:	ac620000	sw	v0,0(v1)
58:	03e00008	jr	ra
5c:	30427fff	andi	v0,v0,0x7fff

```
...
```

```
# objdump -r rand_03.o
```

...

RELOCATION RECORDS FOR [.text]:

OFFSET	TYPE	VALUE
00000000	R_MIPS_HI16	.bss
00000008	R_MIPS_L016	.bss
0000000c	R_MIPS_HI16	.bss
00000010	R_MIPS_L016	.bss
00000054	R_MIPS_L016	.bss

...

20.5

: 65.1 on page 879.

Capítulo 21

Estructuras

21.1 MSVC:

Listing 21.1: WinBase.h

```
typedef struct _SYSTEMTIME {  
    WORD wYear;  
    WORD wMonth;  
    WORD wDayOfWeek;  
    WORD wDay;  
    WORD wHour;  
    WORD wMinute;  
    WORD wSecond;  
    WORD wMilliseconds;  
} SYSTEMTIME, *PSYSTEMTIME;
```

```
#include <windows.h>  
#include <stdio.h>  
  
void main()  
{  
    SYSTEMTIME t;  
    GetSystemTime (&t);  
  
    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",  
           t.wYear, t.wMonth, t.wDay,  
           t.wHour, t.wMinute, t.wSecond);  
  
    return;  
};
```

(MSVC 2010):

Listing 21.2: MSVC 2010 /GS-

```
_t$ = -16 ; size = 16
_main      PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 16
    lea     eax, DWORD PTR _t$[ebp]
    push    eax
    call    DWORD PTR __imp__GetSystemTime@4
    movzx   ecx, WORD PTR _t$[ebp+12] ; wSecond
    push    ecx
    movzx   edx, WORD PTR _t$[ebp+10] ; wMinute
    push    edx
    movzx   eax, WORD PTR _t$[ebp+8]  ; wHour
    push    eax
    movzx   ecx, WORD PTR _t$[ebp+6]  ; wDay
    push    ecx
    movzx   edx, WORD PTR _t$[ebp+2]  ; wMonth
    push    edx
    movzx   eax, WORD PTR _t$[ebp]   ; wYear
    push    eax
    push    OFFSET $SG78811 ; '%04d-%02d-%02d %02d:%02d:%02d', 0x
    ↓ aH, 00H
    call    _printf
    add    esp, 28
    xor    eax, eax
    mov    esp, ebp
    pop    ebp
    ret    0
_main      ENDP
```

21.1.1 OllyDbg

MSVC 2010 /GS- /MD OllyDbg. GetSystemTime(),:

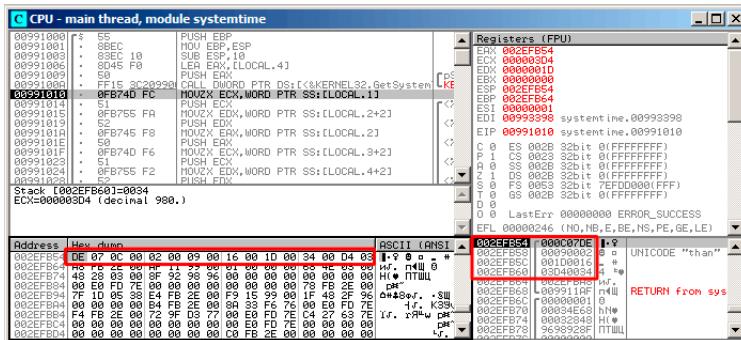


Figura 21.1: OllyDbg: GetSystemTime()

9 2014, 22:29:52:

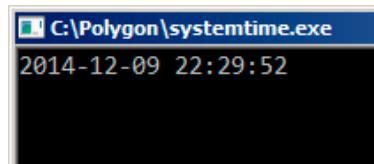


Figura 21.2: OllyDbg:

```
DE 07 0C 00 02 00 09 00 16 00 1D 00 34 00 D4 03
```

... :

0x07DE	2014	wYear
0x000C	12	wMonth
0x0002	2	wDayOfWeek
0x0009	9	wDay
0x0016	22	wHour
0x001D	29	wMinute
0x0034	52	wSecond
0x03D4	980	wMilliseconds

```

printf().
printf() (wDayOfWeek y wMilliseconds),

```

21.1.2

```

#include <windows.h>
#include <stdio.h>

void main()
{
    WORD array[8];
    GetSystemTime (array);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
            array[0] /* wYear */, array[1] /* wMonth */, array[3] /*
            /* wDay */,
            array[4] /* wHour */, array[5] /* wMinute */, array[6] /*
            /* wSecond */);

    return;
}

```

```

systemtime2.c(7) : warning C4133: 'function' : incompatible /*
    types - from 'WORD [8]' to 'LPSYSTEMTIME'

```

:

Listing 21.3: Sin optimización MSVC 2010

```

$SG78573 DB      '%04d-%02d-%02d %02d:%02d:%02d', 0aH, 00H

_array$ = -16    ; size = 16
_main    PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 16
    lea     eax, DWORD PTR _array$[ebp]
    push    eax
    call    DWORD PTR __imp__GetSystemTime@4
    movzx  ecx, WORD PTR _array$[ebp+12] ; wSecond
    push    ecx
    movzx  edx, WORD PTR _array$[ebp+10] ; wMinute
    push    edx
    movzx  eax, WORD PTR _array$[ebp+8] ; wHour
    push    eax

```

```

    movzx  ecx, WORD PTR _array$[ebp+6] ; wDay
    push   ecx
    movzx  edx, WORD PTR _array$[ebp+2] ; wMonth
    push   edx
    movzx  eax, WORD PTR _array$[ebp] ; wYear
    push   eax
    push   OFFSET $SG78573
    call   _printf
    add    esp, 28
    xor    eax, eax
    mov    esp, ebp
    pop    ebp
    ret    0
_main  ENDP
!
```

21.2

```

#include <windows.h>
#include <stdio.h>

void main()
{
    SYSTEMTIME *t;
    t=(SYSTEMTIME *)malloc (sizeof (SYSTEMTIME));
    GetSystemTime (t);
    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
            t->wYear, t->wMonth, t->wDay,
            t->wHour, t->wMinute, t->wSecond);
    free (t);
    return;
}

```

Listing 21.4: Con optimización MSVC

<code>_main</code>	PROC
--------------------	------

```

push    esi
push    16
call    _malloc
add    esp, 4
mov    esi, eax
push    esi
call    DWORD PTR __imp__GetSystemTime@4
movzx  eax, WORD PTR [esi+12] ; wSecond
movzx  ecx, WORD PTR [esi+10] ; wMinute
movzx  edx, WORD PTR [esi+8] ; wHour
push    eax
movzx  eax, WORD PTR [esi+6] ; wDay
push    ecx
movzx  ecx, WORD PTR [esi+2] ; wMonth
push    edx
movzx  edx, WORD PTR [esi] ; wYear
push    eax
push    ecx
push    edx
push    OFFSET $SG78833
call    _printf
push    esi
call    _free
add    esp, 32
xor    eax, eax
pop    esi
ret    0
_main    ENDP

```

```

#include <windows.h>
#include <stdio.h>

void main()
{
    WORD *t;

    t=(WORD *)malloc (16);

    GetSystemTime (t);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
           t[0] /* wYear */, t[1] /* wMonth */, t[3] /* wDay */,
           t[4] /* wHour */, t[5] /* wMinute */, t[6] /* wSecond */);

    free (t);
}

```

```
    return;
};
```

:

Listing 21.5: Con optimización MSVC

```
$SG78594 DB      '%04d-%02d-%02d %02d:%02d:%02d', 0aH, 00H

_main PROC
    push    esi
    push    16
    call    _malloc
    add     esp, 4
    mov     esi, eax
    push    esi
    call    DWORD PTR __imp__GetSystemTime@4
    movzx  eax, WORD PTR [esi+12]
    movzx  ecx, WORD PTR [esi+10]
    movzx  edx, WORD PTR [esi+8]
    push    eax
    movzx  eax, WORD PTR [esi+6]
    push    ecx
    movzx  ecx, WORD PTR [esi+2]
    push    edx
    movzx  edx, WORD PTR [esi]
    push    eax
    push    ecx
    push    edx
    push    OFFSET $SG78594
    call    _printf
    push    esi
    call    _free
    add     esp, 32
    xor     eax, eax
    pop     esi
    ret     0
_main ENDP
```

21.3 UNIX: struct tm

21.3.1 Linux

```
#include <stdio.h>
#include <time.h>
```

```

void main()
{
    struct tm t;
    time_t unix_time;

    unix_time=time(NULL);

    localtime_r (&unix_time, &t);

    printf ("Year: %d\n", t.tm_year+1900);
    printf ("Month: %d\n", t.tm_mon);
    printf ("Day: %d\n", t.tm_mday);
    printf ("Hour: %d\n", t.tm_hour);
    printf ("Minutes: %d\n", t.tm_min);
    printf ("Seconds: %d\n", t.tm_sec);
}

```

GCC 4.4.1:

Listing 21.6: GCC 4.4.1

```

main proc near
    push    ebp
    mov     ebp, esp
    and     esp, 0FFFFFFF0h
    sub     esp, 40h
    mov     dword ptr [esp], 0 ;  time()
    call    time
    mov     [esp+3Ch], eax
    lea     eax, [esp+3Ch]   ;
    lea     edx, [esp+10h]   ;
    mov     [esp+4], edx     ;
    mov     [esp], eax       ;  time()
    call    localtime_r
    mov     eax, [esp+24h]   ; tm_year
    lea     edx, [eax+76Ch] ; edx=eax+1900
    mov     eax, offset format ; "Year: %d\n"
    mov     [esp+4], edx
    mov     [esp], eax
    call    printf
    mov     edx, [esp+20h]   ; tm_mon
    mov     eax, offset aMonthD ; "Month: %d\n"
    mov     [esp+4], edx
    mov     [esp], eax
    call    printf
    mov     edx, [esp+1Ch]   ; tm_mday
    mov     eax, offset aDayD ; "Day: %d\n"

```

```

    mov      [esp+4], edx
    mov      [esp], eax
    call    printf
    mov      edx, [esp+18h]      ; tm_hour
    mov      eax, offset aHourD ; "Hour: %d\n"
    mov      [esp+4], edx
    mov      [esp], eax
    call    printf
    mov      edx, [esp+14h]      ; tm_min
    mov      eax, offset aMinutesD ; "Minutes: %d\n"
    mov      [esp+4], edx
    mov      [esp], eax
    call    printf
    mov      edx, [esp+10h]
    mov      eax, offset aSecondsD ; "Seconds: %d\n"
    mov      [esp+4], edx      ; tm_sec
    mov      [esp], eax
    call    printf
    leave
    retn
main endp

```

GDB

1:

Listing 21.7: GDB

```

dennis@ubuntuvm:~/polygon$ date
Mon Jun  2 18:10:37 EEST 2014
dennis@ubuntuvm:~/polygon$ gcc GCC_tm.c -o GCC_tm
dennis@ubuntuvm:~/polygon$ gdb GCC_tm
GNU gdb (GDB) 7.6.1-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/dennis/polygon/GCC_tm... (no debugging symbols found)...done.

```

```
(gdb) b printf
Breakpoint 1 at 0x8048330
(gdb) run
Starting program: /home/dennis/polygon/GCC_tm

Breakpoint 1, __printf (format=0x80485c0 "Year: %d\n") at ↵
    ↳ printf.c:29
29      printf.c: No such file or directory.
(gdb) x/20x $esp
0xbffff0dc: 0x080484c3      0x080485c0      0x0000007de ↵
    ↳ 0x00000000
0xbffff0ec: 0x08048301      0x538c93ed      0x00000025 ↵
    ↳ 0x0000000a
0xbffff0fc: 0x000000012     0x000000002     0x000000005 ↵
    ↳ 0x00000072
0xbffff10c: 0x000000001     0x000000098     0x000000001 ↵
    ↳ 0x00002a30
0xbffff11c: 0x0804b090     0x08048530     0x000000000 ↵
    ↳ 0x00000000
(gdb)
```

. time.h:

Listing 21.8: time.h

```
struct tm
{
    int    tm_sec;
    int    tm_min;
    int    tm_hour;
    int    tm_mday;
    int    tm_mon;
    int    tm_year;
    int    tm_wday;
    int    tm_yday;
    int    tm_isdst;
};
```

..

:

0xbffff0dc:	0x080484c3	0x080485c0	0x0000007de ↵
↳	0x00000000		
0xbffff0ec:	0x08048301	0x538c93ed	0x00000025 sec ↵
↳	0x0000000a min		
0xbffff0fc:	0x000000012 hour	0x000000002 mday	0x000000005 mon ↵
↳	0x00000072 year		

```

0xbfffff10c: 0x00000001 wday 0x00000098 yday 0x00000001 ↵
    ↳ isdst0x00002a30
0xbfffff11c: 0x0804b090      0x08048530      0x00000000 ↵
    ↳ 0x00000000
:
```

0x00000025	37	tm_sec
0x0000000a	10	tm_min
0x00000012	18	tm_hour
0x00000002	2	tm_mday
0x00000005	5	tm_mon
0x00000072	114	tm_year
0x00000001	1	tm_wday
0x00000098	152	tm_yday
0x00000001	1	tm_isdst

SYSTEMTIME ([21.1 on page 446](#)), tm_wday, tm_yday, tm_isdst.

21.3.2 ARM

Con optimización Keil 6/2013 (Modo Thumb)

:

Listing 21.9: Con optimización Keil 6/2013 (Modo Thumb)

```

var_38 = -0x38
var_34 = -0x34
var_30 = -0x30
var_2C = -0x2C
var_28 = -0x28
var_24 = -0x24
timer  = -0xC

        PUSH    {LR}
        MOVS   R0, #0          ; timer
        SUB    SP, SP, #0x34
        BL     time
        STR    R0, [SP,#0x38+timer]
        MOV    R1, SP          ; tp
        ADD    R0, SP, #0x38+timer ; timer
        BL     localtime_r
        LDR    R1, =0x76C
        LDR    R0, [SP,#0x38+var_24]
        ADDS  R1, R0, R1
:
```

```

ADR    R0, aYearD      ; "Year: %d\n"
BL     __2printf
LDR    R1, [SP,#0x38+var_28]
ADR    R0, aMonthD     ; "Month: %d\n"
BL    __2printf
LDR    R1, [SP,#0x38+var_2C]
ADR    R0, aDayD       ; "Day: %d\n"
BL    __2printf
LDR    R1, [SP,#0x38+var_30]
ADR    R0, aHourD      ; "Hour: %d\n"
BL    __2printf
LDR    R1, [SP,#0x38+var_34]
ADR    R0, aMinutesD   ; "Minutes: %d\n"
BL    __2printf
LDR    R1, [SP,#0x38+var_38]
ADR    R0, aSecondsD   ; "Seconds: %d\n"
BL    __2printf
ADD    SP, SP, #0x34
POP   {PC}

```

Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

IDA «» tm (IDA «» localtime_r()), .

Listing 21.10: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

var_38 = -0x38
var_34 = -0x34

PUSH {R7,LR}
MOV R7, SP
SUB SP, SP, #0x30
MOVS R0, #0 ; time_t *
BLX _time
ADD R1, SP, #0x38+var_34 ; struct tm *
STR R0, [SP,#0x38+var_38]
MOV R0, SP ; time_t *
BLX _localtime_r
LDR R1, [SP,#0x38+var_34.tm_year]
MOV R0, 0xF44 ; "Year: %d\n"
ADD R0, PC ; char *
ADDW R1, R1, #0x76C
BLX _printf
LDR R1, [SP,#0x38+var_34.tm_mon]
MOV R0, 0xF3A ; "Month: %d\n"
ADD R0, PC ; char *
BLX _printf

```

```

LDR R1, [SP,#0x38+var_34.tm_mday]
MOV R0, 0xF35 ; "Day: %d\n"
ADD R0, PC ; char *
BLX _printf
LDR R1, [SP,#0x38+var_34.tm_hour]
MOV R0, 0xF2E ; "Hour: %d\n"
ADD R0, PC ; char *
BLX _printf
LDR R1, [SP,#0x38+var_34.tm_min]
MOV R0, 0xF28 ; "Minutes: %d\n"
ADD R0, PC ; char *
BLX _printf
LDR R1, [SP,#0x38+var_34]
MOV R0, 0xF25 ; "Seconds: %d\n"
ADD R0, PC ; char *
BLX _printf
ADD SP, SP, #0x30
POP {R7,PC}

...
00000000 tm      struc ; (sizeof=0x2C, standard type)
00000000 tm_sec  DCD ?
00000004 tm_min  DCD ?
00000008 tm_hour DCD ?
0000000C tm_mday DCD ?
00000010 tm_mon   DCD ?
00000014 tm_year  DCD ?
00000018 tm_wday  DCD ?
0000001C tm_yday  DCD ?
00000020 tm_isdst DCD ?
00000024 tm_gmtoff DCD ?
00000028 tm_zone   DCD ? ; offset
0000002C tm      ends

```

21.3.3 MIPS

Listing 21.11: Con optimización GCC 4.4.5 (IDA)

```

1 main:
2 ;
3 ;
4 ;
5 ;
6 var_40      = -0x40
7 var_38      = -0x38
8 seconds     = -0x34

```

```

9 minutes      = -0x30
10 hour        = -0x2C
11 day         = -0x28
12 month       = -0x24
13 year        = -0x20
14 var_4        = -4
15
16             lui    $gp, (__gnu_local_gp >> 16)
17             addiu $sp, -0x50
18             la     $gp, (__gnu_local_gp & 0xFFFF)
19             sw     $ra, 0x50+var_4($sp)
20             sw     $gp, 0x50+var_40($sp)
21             lw     $t9, (time & 0xFFFF)($gp)
22             or     $at, $zero ; load delay slot, NOP
23             jalr   $t9
24             move   $a0, $zero ; branch delay slot, NOP
25             lw     $gp, 0x50+var_40($sp)
26             addiu $a0, $sp, 0x50+var_38
27             lw     $t9, (localtime_r & 0xFFFF)($gp)
28             addiu $a1, $sp, 0x50+seconds
29             jalr   $t9
30             sw     $v0, 0x50+var_38($sp) ; branch delay ↴
31             ↵ slot
32             lw     $gp, 0x50+var_40($sp)
33             lw     $a1, 0x50+year($sp)
34             lw     $t9, (printf & 0xFFFF)($gp)
35             la     $a0, $LC0          # "Year: %d\n"
36             jalr   $t9
37             addiu $a1, 1900 ; branch delay slot
38             lw     $gp, 0x50+var_40($sp)
39             lw     $a1, 0x50+month($sp)
40             lw     $t9, (printf & 0xFFFF)($gp)
41             lui    $a0, ($LC1 >> 16) # "Month: %d\n"
42             jalr   $t9
43             la     $a0, ($LC1 & 0xFFFF) # "Month: %d\n" ; ↴
44             ↵ branch delay slot
45             lw     $gp, 0x50+var_40($sp)
46             lw     $a1, 0x50+day($sp)
47             lw     $t9, (printf & 0xFFFF)($gp)
48             lui    $a0, ($LC2 >> 16) # "Day: %d\n"
49             jalr   $t9
50             la     $a0, ($LC2 & 0xFFFF) # "Day: %d\n" ; ↴
51             ↵ branch delay slot
52             lw     $gp, 0x50+var_40($sp)
53             lw     $a1, 0x50+hour($sp)
54             lw     $t9, (printf & 0xFFFF)($gp)
55             lui    $a0, ($LC3 >> 16) # "Hour: %d\n"

```

```

53          jalr    $t9
54          la      $a0, ($LC3 & 0xFFFF) # "Hour: %d\n" ;
55          ↳ branch delay slot
56          lw      $gp, 0x50+var_40($sp)
57          lw      $a1, 0x50+minutes($sp)
58          lw      $t9, (printf & 0xFFFF)($gp)
59          lui    $a0, ($LC4 >> 16) # "Minutes: %d\n"
60          jalr    $t9
61          la      $a0, ($LC4 & 0xFFFF) # "Minutes: %d\n" ;
62          ↳ ; branch delay slot
63          lw      $gp, 0x50+var_40($sp)
64          lw      $a1, 0x50+seconds($sp)
65          lw      $t9, (printf & 0xFFFF)($gp)
66          lui    $a0, ($LC5 >> 16) # "Seconds: %d\n"
67          jalr    $t9
68          la      $a0, ($LC5 & 0xFFFF) # "Seconds: %d\n" ;
69          ↳ ; branch delay slot
70          lw      $ra, 0x50+var_4($sp)
71          or      $at, $zero ; load delay slot, NOP
72          jr      $ra
73          addiu   $sp, 0x50
74
75          $LC0: .ascii "Year: %d\n"<0>
76          $LC1: .ascii "Month: %d\n"<0>
77          $LC2: .ascii "Day: %d\n"<0>
78          $LC3: .ascii "Hour: %d\n"<0>
79          $LC4: .ascii "Minutes: %d\n"<0>
80          $LC5: .ascii "Seconds: %d\n"<0>

```

21.3.4

: Spanish text placeholder.[21.8](#).

```
#include <stdio.h>
#include <time.h>

void main()
{
    int tm_sec, tm_min, tm_hour, tm_mday, tm_mon, tm_year, ↳
    ↳ tm_wday, tm_yday, tm_isdst;
    time_t unix_time;

    unix_time=time(NULL);

    localtime_r (&unix_time, &tm_sec);

    printf ("Year: %d\n", tm_year+1900);
}
```

```

printf ("Month: %d\n", tm_mon);
printf ("Day: %d\n", tm_mday);
printf ("Hour: %d\n", tm_hour);
printf ("Minutes: %d\n", tm_min);
printf ("Seconds: %d\n", tm_sec);
}

```

N.B.

:

Listing 21.12: GCC 4.7.3

```

GCC_tm2.c: In function 'main':
GCC_tm2.c:11:5: warning: passing argument 2 of 'localtime_r' ↴
  ↴ from incompatible pointer type [enabled by default]
In file included from GCC_tm2.c:2:0:
/usr/include/time.h:59:12: note: expected 'struct tm *' but ↴
  ↴ argument is of type 'int *'

```

:

Listing 21.13: GCC 4.7.3

```

main      proc near

var_30     = dword ptr -30h
var_2C     = dword ptr -2Ch
unix_time  = dword ptr -1Ch
tm_sec     = dword ptr -18h
tm_min     = dword ptr -14h
tm_hour    = dword ptr -10h
tm_mday    = dword ptr -0Ch
tm_mon     = dword ptr -8
tm_year    = dword ptr -4

push      ebp
mov       ebp, esp
and       esp, 0FFFFFFF0h
sub       esp, 30h
call      __main
mov       [esp+30h+var_30], 0 ; arg 0
call      time
mov       [esp+30h+unix_time], eax
lea        eax, [esp+30h+tm_sec]
mov       [esp+30h+var_2C], eax
lea        eax, [esp+30h+unix_time]
mov       [esp+30h+var_30], eax
call      localtime_r

```

```

        mov    eax, [esp+30h+tm_year]
        add    eax, 1900
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aYearD ; "Year: %d\n"
        ↵ "
        call   printf
        mov    eax, [esp+30h+tm_mon]
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aMonthD ; "Month: %d\n"
        ↵ \n"
        call   printf
        mov    eax, [esp+30h+tm_mday]
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aDayD ; "Day: %d\n"
        call   printf
        mov    eax, [esp+30h+tm_hour]
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aHourD ; "Hour: %d\n"
        ↵ "
        call   printf
        mov    eax, [esp+30h+tm_min]
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aMinutesD ; "Minutes\n"
        ↵ : %d\n"
        call   printf
        mov    eax, [esp+30h+tm_sec]
        mov    [esp+30h+var_2C], eax
        mov    [esp+30h+var_30], offset aSecondsD ; "Seconds\n"
        ↵ : %d\n"
        call   printf
        leave
        retn
main    endp

```

..
`tm_year, tm_mon, tm_mday, tm_hour, tm_min tm_sec . localtime_r(). GetSystemTime() () . : «Organización de campos en la estructura» (21.4 on page 466).`

[Rit93].

21.3.5

```
#include <stdio.h>
#include <time.h>
```

```

void main()
{
    struct tm t;
    time_t unix_time;
    int i;

    unix_time=time(NULL);

    localtime_r (&unix_time, &t);

    for (i=0; i<9; i++)
    {
        int tmp=((int*)&t)[i];
        printf ("0x%08X (%d)\n", tmp, tmp);
    };
}

```

.! 23:51:45 26-July-2014.

```

0x00000002D (45)
0x000000033 (51)
0x000000017 (23)
0x00000001A (26)
0x000000006 (6)
0x000000072 (114)
0x000000006 (6)
0x0000000CE (206)
0x000000001 (1)

```

: 21.8 on page 455.

:

Listing 21.14: Con optimización GCC 4.8.1

```

main          proc near
             push   ebp
             mov    ebp, esp
             push   esi
             push   ebx
             and    esp, 0FFFFFFF0h
             sub    esp, 40h
             mov    dword ptr [esp], 0 ; timer
             lea    ebx, [esp+14h]
             call   _time
             lea    esi, [esp+38h]
             mov    [esp+4], ebx      ; tp

```

```

        mov    [esp+10h], eax
        lea    eax, [esp+10h]
        mov    [esp], eax      ; timer
        call   _localtime_r
        nop
        lea    esi, [esi+0]    ; NOP
loc_80483D8:
; .
        mov    eax, [ebx]      ;
        add    ebx, 4          ;
        mov    dword ptr [esp+4], offset a0x08xD ; "0x2
        ↳ %08X (%d)\n"
        mov    dword ptr [esp], 1
        mov    [esp+0Ch], eax ; printf()
        mov    [esp+8], eax ; printf()
        call   __printf_chk
        cmp    ebx, esi        ; ?
        jnz   short loc_80483D8      ;
        lea    esp, [ebp-8]
        pop    ebx
        pop    esi
        pop    ebp
        retn
main
endp

```

Ejercicio

21.3.6

```

#include <stdio.h>
#include <time.h>

void main()
{
    struct tm t;
    time_t unix_time;
    int i, j;

    unix_time=time(NULL);

    localtime_r (&unix_time, &t);

    for (i=0; i<9; i++)
    {
        for (j=0; j<4; j++)

```

```

        printf ("0x%02X ", ((unsigned char*)&t)[i*4+j]);
        printf ("\n");
    };
}
};
```

```

0x2D 0x00 0x00 0x00
0x33 0x00 0x00 0x00
0x17 0x00 0x00 0x00
0x1A 0x00 0x00 0x00
0x06 0x00 0x00 0x00
0x72 0x00 0x00 0x00
0x06 0x00 0x00 0x00
0xCE 0x00 0x00 0x00
0x01 0x00 0x00 0x00
```

23:51:45 26-July-2014 ². ([21.3.5 on page 463](#)), ([31 on page 586](#)).

Listing 21.15: Con optimización GCC 4.8.1

```

main          proc near
    push    ebp
    mov     ebp, esp
    push    edi
    push    esi
    push    ebx
    and    esp, 0FFFFFFF0h
    sub    esp, 40h
    mov    dword ptr [esp], 0 ; timer
    lea    esi, [esp+14h]
    call   _time
    lea    edi, [esp+38h] ; struct end
    mov    [esp+4], esi ; tp
    mov    [esp+10h], eax
    lea    eax, [esp+10h]
    mov    [esp], eax ; timer
    call   _localtime_r
    lea    esi, [esi+0] ; NOP
;
loc_8048408: xor    ebx, ebx ; j=0
loc_804840A: movzx  eax, byte ptr [esi+ebx] ;
                add    ebx, 1 ; j=j+1
```

```

        mov    dword ptr [esp+4], offset a0x02x ; "0x2
        ↳ %02X "
        mov    dword ptr [esp], 1
        mov    [esp+8], eax      ; printf()
        call   __printf_chk
        cmp    ebx, 4
        jnz    short loc_804840A
; (CR)
        mov    dword ptr [esp], 0Ah ; c
        add    esi, 4
        call   _putchar
        cmp    esi, edi      ; ?
        jnz    short loc_8048408 ; j=0
        lea    esp, [ebp-0Ch]
        pop    ebx
        pop    esi
        pop    edi
        pop    ebp
        retn
main    endp

```

21.4 Organización de campos en la estructura

```

#include <stdio.h>

struct s
{
    char a;
    int b;
    char c;
    int d;
};

void f(struct s s)
{
    printf ("a=%d; b=%d; c=%d; d=%d\n", s.a, s.b, s.c, s.d);
}

int main()
{
    struct s tmp;
    tmp.a=1;
    tmp.b=2;
    tmp.c=3;
    tmp.d=4;
    f(tmp);
}

```

{};

21.4.1 x86

Listing 21.16: MSVC 2012 /GS- /Ob0

```
1 _tmp$ = -16
2
3 _main    PROC
4     push    ebp
5     mov     ebp, esp
6     sub     esp, 16
7     mov     BYTE PTR _tmp$[ebp], 1      ; a
8     mov     DWORD PTR _tmp$[ebp+4], 2   ; b
9     mov     BYTE PTR _tmp$[ebp+8], 3   ; c
10    mov    DWORD PTR _tmp$[ebp+12], 4 ; d
11    sub    esp, 16
12    mov    eax, esp
13    mov    ecx, DWORD PTR _tmp$[ebp]  ;
14    mov    DWORD PTR [eax], ecx
15    mov    edx, DWORD PTR _tmp$[ebp+4]
16    mov    DWORD PTR [eax+4], edx
17    mov    ecx, DWORD PTR _tmp$[ebp+8]
18    mov    DWORD PTR [eax+8], ecx
19    mov    edx, DWORD PTR _tmp$[ebp+12]
20    mov    DWORD PTR [eax+12], edx
21    call   _f
22    add    esp, 16
23    xor    eax, eax
24    mov    esp, ebp
25    pop    ebp
26    ret    0
27 _main    ENDP
28
29 _s$ = 8 ; size = 16
30 ?f@@YAXUs@@@Z PROC ; f
31     push    ebp
32     mov     ebp, esp
33     mov     eax, DWORD PTR _s$[ebp+12]
34     push    eax
35     movsx  ecx, BYTE PTR _s$[ebp+8]
36     push    ecx
37     mov     edx, DWORD PTR _s$[ebp+4]
38     push    edx
39     movsx  eax, BYTE PTR _s$[ebp]
40     push    eax
41     push    OFFSET $SG3842
```

```

42    call   _printf
43    add    esp, 20
44    pop    ebp
45    ret    0
46 ?f@@YAXUs@@@Z ENDP ; f
47 _TEXT    ENDS

```

(/Zp1) (/Zp[n] pack structures on n-byte boundary).

Listing 21.17: MSVC 2012 /GS- /Zp1

```

1 _main    PROC
2     push   ebp
3     mov    ebp, esp
4     sub    esp, 12
5     mov    BYTE PTR _tmp$[ebp], 1      ; a
6     mov    DWORD PTR _tmp$[ebp+1], 2    ; b
7     mov    BYTE PTR _tmp$[ebp+5], 3      ; c
8     mov    DWORD PTR _tmp$[ebp+6], 4    ; d
9
10    sub   esp, 12
11    mov   eax, esp
12    mov   ecx, DWORD PTR _tmp$[ebp]
13    mov   DWORD PTR [eax], ecx
14    mov   edx, DWORD PTR _tmp$[ebp+4]
15    mov   DWORD PTR [eax+4], edx
16    mov   cx, WORD PTR _tmp$[ebp+8]
17    mov   WORD PTR [eax+8], cx
18    call   _f
19    add   esp, 12
20    xor   eax, eax
21    mov   esp, ebp
22    pop   ebp
23    ret   0
24 _main    ENDP
25
26 _TEXT    SEGMENT
27 _s$ = 8 ; size = 10
28 ?f@@YAXUs@@@Z PROC      ; f
29     push   ebp
30     mov    ebp, esp
31     mov    eax, DWORD PTR _s$[ebp+6]
32     push   eax
33     movsx  ecx, BYTE PTR _s$[ebp+5]
34     push   ecx
35     mov    edx, DWORD PTR _s$[ebp+1]
36     push   edx
37     movsx  eax, BYTE PTR _s$[ebp]

```

```

38     push    eax
39     push    OFFSET $SG3842
40     call    _printf
41     add     esp, 20
42     pop     ebp
43     ret     0
44 ?f@@YAXUs@@@Z ENDP      ; f

```

Listing 21.18: WinNT.h

```
#include "pshpack1.h"
```

Listing 21.19: WinNT.h

```
#include "pshpack4.h"                                // 4 byte packing is ↴
   ↴ the default
```

Listing 21.20: PshPack1.h

```

#if ! (defined(lint) || defined(RC_INVOKED))
#if ( _MSC_VER >= 800 && !defined(_M_I86)) || defined( ↴
   ↴ _PUSHPOP_SUPPORTED)
#pragma warning(disable:4103)
#if !(defined( MIDL_PASS )) || defined( __midl )
#pragma pack(push,1)
#else
#pragma pack(1)
#endif
#else
#pragma pack(1)
#endif
#endif /* ! (defined(lint) || defined(RC_INVOKED)) */

```

OllyDbg +

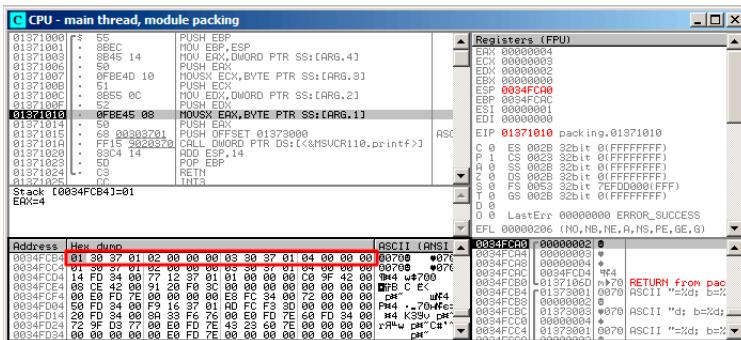


Figura 21.3: OllyDbg:

. : Spanish text placeholder.[21.16](#) (34 y 38).

```
unsigned char uint8_t, .
```

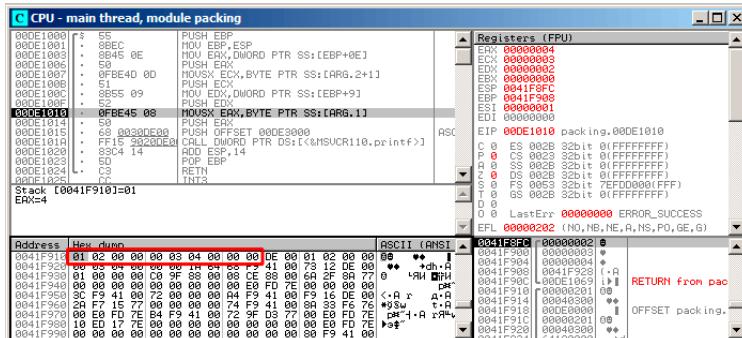
OllyDbg +

Figura 21.4: OllyDbg:

21.4.2 ARM**Con optimización Keil 6/2013 (Modo Thumb)**

Listing 21.21: Con optimización Keil 6/2013 (Modo Thumb)

```

.text:0000003E          exit ; CODE XREF: f+16
.text:0000003E 05 B0      ADD     SP, SP, #0x14
.text:00000040 00 BD      POP    {PC}

.text:00000280          f
.text:00000280          var_18 = -0x18
.text:00000280          a      = -0x14
.text:00000280          b      = -0x10
.text:00000280          c      = -0xC
.text:00000280          d      = -8

.text:00000280 0F B5      PUSH   {R0-R3,LR}
.text:00000282 81 B0      SUB    SP, SP, #4
.text:00000284 04 98      LDR    R0, [SP,#16] ; d
.text:00000286 02 9A      LDR    R2, [SP,#8]  ; b
.text:00000288 00 90      STR    R0, [SP]
.text:0000028A 68 46      MOV    R0, SP
.text:0000028C 03 7B      LDRB   R3, [R0,#12] ; c
.text:0000028E 01 79      LDRB   R1, [R0,#4]  ; a
.text:00000290 59 A0      ADR    R0, aADBDCDDD ; "a"
    ↴ =%d; b=%d; c=%d; d=%d\n"
.text:00000292 05 F0 AD FF      BL    __2printf

```

.text:00000296 D2 E6	B	exit
----------------------	---	------

R0-R3.

LDRB MOVSX x86. *a* y *c*.

($(5 * 4 = 0x14)$).

ARM + Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

Listing 21.22: Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```
var_C = -0xC
```

```
PUSH {R7,LR}
MOV R7, SP
SUB SP, SP, #4
MOV R9, R1 ; b
MOV R1, R0 ; a
MOVW R0, #0xF10 ; "a=%d; b=%d; c=%d; d=%d\n"
SXTB R1, R1 ; prepare a
MOVT.W R0, #0
STR R3, [SP,#0xC+var_C] ; place d to stack for printf
()
ADD R0, PC ; format-string
SXTB R3, R2 ; prepare c
MOV R2, R9 ; b
BLX _printf
ADD SP, SP, #4
POP {R7,PC}
```

SXTB (*Signed Extend Byte*) MOVSX en x86.

21.4.3 MIPS

Listing 21.23: Con optimización GCC 4.4.5 (IDA)

```
1 f:
2
3 var_18      = -0x18
4 var_10      = -0x10
5 var_4       = -4
6 arg_0       = 0
7 arg_4       = 4
8 arg_8       = 8
9 arg_C       = 0xC
10
```

```

11 ; $a0=s.a
12 ; $a1=s.b
13 ; $a2=s.c
14 ; $a3=s.d
15         lui      $gp, (__gnu_local_gp >> 16)
16         addiu   $sp, -0x28
17         la       $gp, (__gnu_local_gp & 0xFFFF)
18         sw       $ra, 0x28+var_4($sp)
19         sw       $gp, 0x28+var_10($sp)
20 ; prepare byte from 32-bit big-endian integer:
21         sra     $t0, $a0, 24
22         move    $v1, $a1
23 ; prepare byte from 32-bit big-endian integer:
24         sra     $v0, $a2, 24
25         lw      $t9, (printf & 0xFFFF)($gp)
26         sw      $a0, 0x28+arg_0($sp)
27         lui     $a0, ($LC0 >> 16) # "a=%d; b=%d; c=%d; \n"
28         ↳ d=%d\n"
29         sw      $a3, 0x28+var_18($sp)
30         sw      $a1, 0x28+arg_4($sp)
31         sw      $a2, 0x28+arg_8($sp)
32         sw      $a3, 0x28+arg_C($sp)
33         la      $a0, ($LC0 & 0xFFFF) # "a=%d; b=%d; c=%d; \n"
34         ↳ =%d; d=%d\n"
35         move   $a1, $t0
36         move   $a2, $v1
37         jalr   $t9
38         move   $a3, $v0 ; branch delay slot
39         lw      $ra, 0x28+var_4($sp)
40         or      $at, $zero ; load delay slot, NOP
41         jr      $ra
42         addiu $sp, 0x28 ; branch delay slot
42 $LC0: .ascii "a=%d; b=%d; c=%d; d=%d\n"<0>

```

?

21.4.4

```

void f(char a, int b, char c, int d)
{
    printf ("a=%d; b=%d; c=%d; d=%d\n", a, b, c, d);
}

```

21.5

```
#include <stdio.h>

struct inner_struct
{
    int a;
    int b;
};

struct outer_struct
{
    char a;
    int b;
    struct inner_struct c;
    char d;
    int e;
};

void f(struct outer_struct s)
{
    printf ("a=%d; b=%d; c.a=%d; c.b=%d; d=%d; e=%d\n",
           s.a, s.b, s.c.a, s.c.b, s.d, s.e);
}

int main()
{
    struct outer_struct s;
    s.a=1;
    s.b=2;
    s.c.a=100;
    s.c.b=101;
    s.d=3;
    s.e=4;
    f(s);
}
```

...

(MSVC 2010):

Listing 21.24: Con optimización MSVC 2010 /O_b0

\$SG2802 DB	'a=%d; b=%d; c.a=%d; c.b=%d; d=%d; e=%d', 0aH, 00	✓
↳ H		
_TEXT	SEGMENT	
_s\$ = 8		

```

_f    PROC
    mov    eax, DWORD PTR _s$[esp+16]
    movsx  ecx, BYTE PTR _s$[esp+12]
    mov    edx, DWORD PTR _s$[esp+8]
    push   eax
    mov    eax, DWORD PTR _s$[esp+8]
    push   ecx
    mov    ecx, DWORD PTR _s$[esp+8]
    push   edx
    movsx  edx, BYTE PTR _s$[esp+8]
    push   eax
    push   ecx
    push   edx
    push   OFFSET $SG2802 ; 'a=%d; b=%d; c.a=%d; c.b=%d; d=%d; '
    ↓ e=%d'
    call   _printf
    add    esp, 28
    ret    0
_f    ENDP

_s$ = -24
_main  PROC
    sub    esp, 24
    push   ebx
    push   esi
    push   edi
    mov    ecx, 2
    sub    esp, 24
    mov    eax, esp
    mov    BYTE PTR _s$[esp+60], 1
    mov    ebx, DWORD PTR _s$[esp+60]
    mov    DWORD PTR [eax], ebx
    mov    DWORD PTR [eax+4], ecx
    lea    edx, DWORD PTR [ecx+98]
    lea    esi, DWORD PTR [ecx+99]
    lea    edi, DWORD PTR [ecx+2]
    mov    DWORD PTR [eax+8], edx
    mov    BYTE PTR _s$[esp+76], 3
    mov    ecx, DWORD PTR _s$[esp+76]
    mov    DWORD PTR [eax+12], esi
    mov    DWORD PTR [eax+16], ecx
    mov    DWORD PTR [eax+20], edi
    call   _f
    add    esp, 24
    pop    edi
    pop    esi
    xor    eax, eax

```

```
pop    ebx
add    esp, 24
ret    0
_main  ENDP
```

21.5.1 OllyDbg

OllyDbg outer_struct:

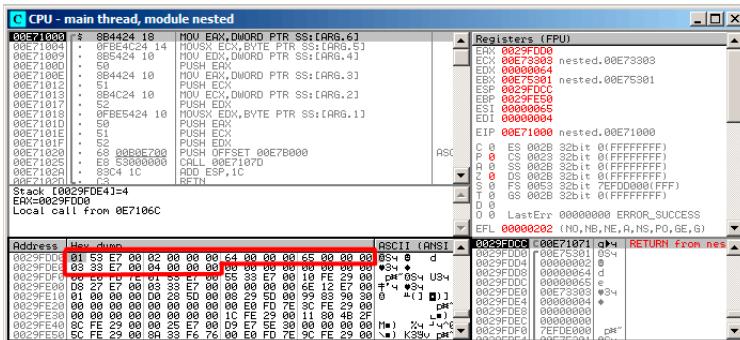


Figura 21.5: OllyDbg:

- *(outer_struct.a)* ;
 - *(outer_struct.b)* () 2;
 - *(inner_struct.a)* () 0x64 (100);
 - *(inner_struct.b)* () 0x65 (101);
 - *(outer_struct.d)* ;
 - *(outer_struct.e)* () 4.

21.6

21.6.1

3:0 (4 Spanish text placeholder)	Stepping
7:4 (4 Spanish text placeholder)	Model
11:8 (4 Spanish text placeholder)	Family
13:12 (2 Spanish text placeholder)	Processor Type
19:16 (4 Spanish text placeholder)	Extended Model
27:20 (8 Spanish text placeholder)	Extended Family

```
#include <stdio.h>
```

```

static inline void cpuid(int code, int *a, int *b, int *c, int *
    ↴ *d) {
    asm volatile("cpuid": "=a"(*a), "=b"(*b), "=c"(*c), "=d"(*d): "a"( ↴
        ↴ code));
}
#endif

#ifndef _MSC_VER
#include <intrin.h>
#endif

struct CPUID_1_EAX
{
    unsigned int stepping:4;
    unsigned int model:4;
    unsigned int family_id:4;
    unsigned int processor_type:2;
    unsigned int reserved1:2;
    unsigned int extended_model_id:4;
    unsigned int extended_family_id:8;
    unsigned int reserved2:4;
};

int main()
{
    struct CPUID_1_EAX *tmp;
    int b[4];

#ifndef _MSC_VER
    __cpuid(b,1);
#endif

#ifndef __GNUC__
    cpuid (1, &b[0], &b[1], &b[2], &b[3]);
#endif

    tmp=(struct CPUID_1_EAX *)&b[0];

    printf ("stepping=%d\n", tmp->stepping);
    printf ("model=%d\n", tmp->model);
    printf ("family_id=%d\n", tmp->family_id);
    printf ("processor_type=%d\n", tmp->processor_type);
    printf ("extended_model_id=%d\n", tmp->extended_model_id);
    printf ("extended_family_id=%d\n", tmp->extended_family_id),
    ↴ ;

    return 0;
}

```

{};

MSVC

:

Listing 21.25: Con optimización MSVC 2008

```
_b$ = -16 ; size = 16
_main    PROC
    sub     esp, 16
    push    ebx

    xor     ecx, ecx
    mov     eax, 1
    cpuid
    push    esi
    lea     esi, DWORD PTR _b$[esp+24]
    mov     DWORD PTR [esi], eax
    mov     DWORD PTR [esi+4], ebx
    mov     DWORD PTR [esi+8], ecx
    mov     DWORD PTR [esi+12], edx

    mov     esi, DWORD PTR _b$[esp+24]
    mov     eax, esi
    and    eax, 15
    push    eax
    push    OFFSET $SG15435 ; 'stepping=%d', 0aH, 00H
    call    _printf

    mov     ecx, esi
    shr    ecx, 4
    and    ecx, 15
    push    ecx
    push    OFFSET $SG15436 ; 'model=%d', 0aH, 00H
    call    _printf

    mov     edx, esi
    shr    edx, 8
    and    edx, 15
    push    edx
    push    OFFSET $SG15437 ; 'family_id=%d', 0aH, 00H
    call    _printf

    mov     eax, esi
    shr    eax, 12
    and    eax, 3
```

```
push    eax
push    OFFSET $SG15438 ; 'processor_type=%d', 0aH, 00H
call    _printf

mov     ecx, esi
shr     ecx, 16
and     ecx, 15
push    ecx
push    OFFSET $SG15439 ; 'extended_model_id=%d', 0aH, 00H
call    _printf

shr     esi, 20
and     esi, 255
push    esi
push    OFFSET $SG15440 ; 'extended_family_id=%d', 0aH, 00H
call    _printf
add    esp, 48
pop    esi

xor    eax, eax
pop    ebx

add    esp, 16
ret    0
_main  ENDP
```

MSVC + OllyDbg

OllyDbg CPUID:

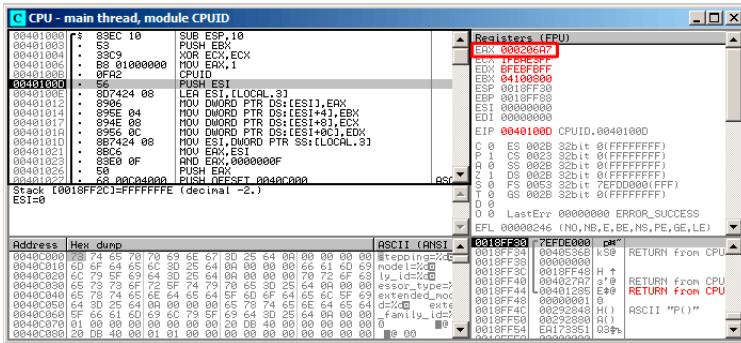


Figura 21.6: OllyDbg:

0x000206A7 (CPU Intel Xeon E3-1220).

00000000000000100000011010100111.

reserved2	0000	0
extended_family_id	00000000	0
extended_model_id	0010	2
reserved1	00	0
processor_id	00	0
family_id	0110	6
model	1010	10
stepping	0111	7

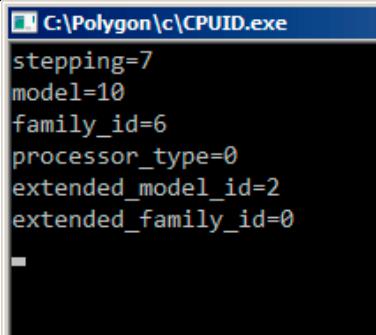


Figura 21.7: OllyDbg:

GCC

Listing 21.26: Con optimización GCC 4.4.1

```

main          proc near ; DATA XREF: _start+17
    push    ebp
    mov     ebp, esp
    and     esp, 0FFFFFFF0h
    push    esi
    mov     esi, 1
    push    ebx
    mov     eax, esi
    sub     esp, 18h
    cpuid
    mov     esi, eax
    and     eax, 0Fh
    mov     [esp+8], eax
    mov     dword ptr [esp+4], offset aSteppingD ; "stepping=%d\n"
    mov     dword ptr [esp], 1
    call    __printf_chk
    mov     eax, esi
    shr     eax, 4
    and     eax, 0Fh
    mov     [esp+8], eax
    mov     dword ptr [esp+4], offset aModelD ; "model=%d\n"
    mov     dword ptr [esp], 1
    call    __printf_chk
    mov     eax, esi
    shr     eax, 8
    and     eax, 0Fh
    mov     [esp+8], eax

```

```

mov    dword ptr [esp+4], offset aFamily_idD ; "family_id\n"
        ↴ %d\n"
mov    dword ptr [esp], 1
call   __printf_chk
mov    eax, esi
shr    eax, 0Ch
and    eax, 3
mov    [esp+8], eax
mov    dword ptr [esp+4], offset aProcessor_type ; "processor_type=%d\n"
        ↴ processor_type=%d\n"
mov    dword ptr [esp], 1
call   __printf_chk
mov    eax, esi
shr    eax, 10h
shr    esi, 14h
and    eax, 0Fh
and    esi, 0FFh
mov    [esp+8], eax
mov    dword ptr [esp+4], offset aExtended_model ; "extended_model_id=%d\n"
        ↴ extended_model_id=%d\n"
mov    dword ptr [esp], 1
call   __printf_chk
mov    [esp+8], esi
mov    dword ptr [esp+4], offset unk_80486D0
mov    dword ptr [esp], 1
call   __printf_chk
add   esp, 18h
xor   eax, eax
pop   ebx
pop   esi
mov   esp, ebp
pop   ebp
retn
main           endp

```

21.6.2 Trabajando con el tipo float como una estructura



(SSpanish text placeholder)

```
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
```

```
#include <memory.h>

struct float_as_struct
{
    unsigned int fraction : 23; // 
    unsigned int exponent : 8;   // + 0x3FF
    unsigned int sign : 1;      // 
};

float f(float )
{
    float f=;
    struct float_as_struct t;

    assert (sizeof (struct float_as_struct) == sizeof (float));

    memcpy (&t, &f, sizeof (float));

    t.sign=1; //
    t.exponent=t.exponent+2; //

    memcpy (&f, &t, sizeof (float));

    return f;
}

int main()
{
    printf ("%f\n", f(1.234));
}
```

Listing 21.27: Sin optimización MSVC 2008

```
_t$ = -8    ; size = 4
_f$ = -4    ; size = 4
__in$ = 8   ; size = 4
?f@@YAMM@Z PROC ; f
    push    ebp
    mov     ebp, esp
    sub     esp, 8

    fld     DWORD PTR __in$[ebp]
    fstp    DWORD PTR _f$[ebp]

    push    4
    lea     eax, DWORD PTR _f$[ebp]
    push    eax
```

```

    lea    ecx, DWORD PTR _t$[ebp]
    push   ecx
    call   _memcpy
    add    esp, 12

    mov    edx, DWORD PTR _t$[ebp]
    or     edx, -2147483648 ; 80000000H -
    mov    DWORD PTR _t$[ebp], edx

    mov    eax, DWORD PTR _t$[ebp]
    shr    eax, 23           ; 00000017H -
    and    eax, 255          ; 000000ffH -
    add    eax, 2             ;
    and    eax, 255          ; 000000ffH
    shl    eax, 23           ; 00000017H -
    mov    ecx, DWORD PTR _t$[ebp]
    and    ecx, -2139095041 ; 807fffffH -

;

    or     ecx, eax
    mov    DWORD PTR _t$[ebp], ecx

    push   4
    lea    edx, DWORD PTR _t$[ebp]
    push   edx
    lea    eax, DWORD PTR _f$[ebp]
    push   eax
    call   _memcpy
    add    esp, 12

    fld    DWORD PTR _f$[ebp]

    mov    esp, ebp
    pop    ebp
    ret    0
?f@@YAMM@Z ENDP      ; f

```

Listing 21.28: Con optimización GCC 4.4.1

```

; f(float)
    public _Z1ff
_Z1ff  proc near

var_4  = dword ptr -4
arg_0  = dword ptr  8

    push    ebp

```

```

    mov    ebp, esp
    sub    esp, 4
    mov    eax, [ebp+arg_0]
    or     eax, 80000000h ;
    mov    edx, eax
    and    eax, 807FFFFFh ;
    shr    edx, 23
    add    edx, 2
    movzx  edx, dl
    shl    edx, 23
    or     eax, edx
    mov    [ebp+var_4], eax
    fld    [ebp+var_4]
    leave
    retn
_Z1ff  endp

main   public main
main   proc near
        push   ebp
        mov    ebp, esp
        and    esp, 0FFFFFFF0h
        sub    esp, 10h
        fld    ds:dword_8048614 ; -4.936
        fstp   qword ptr [esp+8]
        mov    dword ptr [esp+4], offset asc_8048610 ; "%f\n"
        mov    dword ptr [esp], 1
        call   __printf_chk
        xor    eax, eax
        leave
        retn
main   endp

```

21.7 Ejercicios

21.7.1 Ejercicio #1

[Linux x86 \(beginners.re\)³](#)

[Linux MIPS \(beginners.re\)⁴](#)

Responda: [G.1.14 on page 1266.](#)

³GCC 4.8.1 -O3

⁴GCC 4.4.5 -O3

21.7.2 Ejercicio #2

Listing 21.29: Con optimización MSVC 2010

```

$SG2802 DB      '%f', 0aH, 00H
$SG2803 DB      '%c, %d', 0aH, 00H
$SG2805 DB      'error #2', 0aH, 00H
$SG2807 DB      'error #1', 0aH, 00H

__real@405ec000000000000 DQ 0405ec000000000000r ; 123
__real@407bc000000000000 DQ 0407bc000000000000r ; 444

_s$ = 8
_f PROC
    push    esi
    mov     esi, DWORD PTR _s$[esp]
    cmp     DWORD PTR [esi], 1000
    jle     SHORT $LN4@f
    cmp     DWORD PTR [esi+4], 10
    jbe     SHORT $LN3@f
    fld     DWORD PTR [esi+8]
    sub     esp, 8
    fmul   QWORD PTR __real@407bc000000000000
    fld     QWORD PTR [esi+16]
    fmul   QWORD PTR __real@405ec000000000000
    faddp  ST(1), ST(0)
    fstp   QWORD PTR [esp]
    push    OFFSET $SG2802 ; '%f'
    call    _printf
    movzx  eax, BYTE PTR [esi+25]
    movsx  ecx, BYTE PTR [esi+24]
    push    eax
    push    ecx
    push    OFFSET $SG2803 ; '%c, %d'
    call    _printf
    add    esp, 24
    pop    esi
    ret    0

$LN3@f:
    pop    esi
    mov    DWORD PTR _s$[esp-4], OFFSET $SG2805 ; 'error #2'
    jmp    _printf

$LN4@f:
    pop    esi
    mov    DWORD PTR _s$[esp-4], OFFSET $SG2807 ; 'error #1'
    jmp    _printf

```

_f ENDP

Listing 21.30: Sin optimización Keil 6/2013 (Modo ARM)

```

f PROC
    PUSH    {r4-r6,lr}
    MOV     r4,r0
    LDR     r0,[r0,#0]
    CMP     r0,#0x3e8
    ADRLE   r0,|L0.140|
    BLE    |L0.132|
    LDR     r0,[r4,#4]
    CMP     r0,#0xa
    ADRLS   r0,|L0.152|
    BLS    |L0.132|
    ADD     r0,r4,#0x10
    LDM     r0,{r0,r1}
    LDR     r3,|L0.164|
    MOV     r2,#0
    BL      __aeabi_dmul
    MOV     r5,r0
    MOV     r6,r1
    LDR     r0,[r4,#8]
    LDR     r1,|L0.168|
    BL      __aeabi_fmul
    BL      __aeabi_f2d
    MOV     r2,r5
    MOV     r3,r6
    BL      __aeabi_dadd
    MOV     r2,r0
    MOV     r3,r1
    ADR     r0,|L0.172|
    BL      __2printf
    LDRB   r2,[r4,#0x19]
    LDRB   r1,[r4,#0x18]
    POP    {r4-r6,lr}
    ADR     r0,|L0.176|
    B       __2printf
|L0.132|
    POP    {r4-r6,lr}
    B       __2printf
ENDP

|L0.140|
    DCB    "error #1\n",0
    DCB    0
    DCB    0
|L0.152|

```

```

    DCB      "error #2\n",0
    DCB      0
    DCB      0
|L0.164|
    DCD      0x405ec000
|L0.168|
    DCD      0x43de0000
|L0.172|
    DCB      "%f\n",0
|L0.176|
    DCB      "%c, %d\n",0

```

Listing 21.31: Sin optimización Keil 6/2013 (Modo Thumb)

```

f PROC
    PUSH    {r4-r6,lr}
    MOV     r4,r0
    LDR     r0,[r0,#0]
    CMP     r0,#0x3e8
    ADRLE   r0,|L0.140|
    BLE    |L0.132|
    LDR     r0,[r4,#4]
    CMP     r0,#0xa
    ADRLS   r0,|L0.152|
    BLS    |L0.132|
    ADD     r0,r4,#0x10
    LDM     r0,{r0,r1}
    LDR     r3,|L0.164|
    MOV     r2,#0
    BL      __aeabi_dmul
    MOV     r5,r0
    MOV     r6,r1
    LDR     r0,[r4,#8]
    LDR     r1,|L0.168|
    BL      __aeabi_fmul
    BL      __aeabi_f2d
    MOV     r2,r5
    MOV     r3,r6
    BL      __aeabi_dadd
    MOV     r2,r0
    MOV     r3,r1
    ADR     r0,|L0.172|
    BL      __2printf
    LDRB   r2,[r4,#0x19]
    LDRB   r1,[r4,#0x18]
    POP    {r4-r6,lr}
    ADR     r0,|L0.176|
    B      __2printf

```

```

|L0.132|
    POP      {r4-r6,lr}
    B       __2printf
    ENDP

|L0.140|
    DCB      "error #1\n",0
    DCB      0
    DCB      0

|L0.152|
    DCB      "error #2\n",0
    DCB      0
    DCB      0

|L0.164|
    DCD      0x405ec000

|L0.168|
    DCD      0x43de0000

|L0.172|
    DCB      "%f\n",0

|L0.176|
    DCB      "%c, %d\n",0

```

Listing 21.32: Con optimización GCC 4.9 (ARM64)

```

f:
    stp      x29, x30, [sp, -32]!
    add      x29, sp, 0
    ldr      w1, [x0]
    str      x19, [sp,16]
    cmp      w1, 1000
    ble      .L2
    ldr      w1, [x0,4]
    cmp      w1, 10
    bls      .L3
    ldr      s1, [x0,8]
    mov      x19, x0
    ldr      s0, .LC1
    adrp     x0, .LC0
    ldr      d2, [x19,16]
    add      x0, x0, :lo12:.LC0
    fmul    s1, s1, s0
    ldr      d0, .LC2
    fmul    d0, d2, d0
    fcvt    d1, s1
    fadd    d0, d1, d0
    b1      printf
    ldrb    w1, [x19,24]
    adrp     x0, .LC3

```

```

ldr w2, [x19,25]
add x0, x0, :lo12:.LC3
ldr x19, [sp,16]
ldp x29, x30, [sp], 32
b printf
.L3:
ldr x19, [sp,16]
adrp x0, .LC4
ldp x29, x30, [sp], 32
add x0, x0, :lo12:.LC4
b puts
.L2:
ldr x19, [sp,16]
adrp x0, .LC5
ldp x29, x30, [sp], 32
add x0, x0, :lo12:.LC5
b puts
.size f, .-f
.LC1:
.word 1138622464
.LC2:
.word 0
.word 1079951360
.LC0:
.string "%f\n"
.LC3:
.string "%c, %d\n"
.LC4:
.string "error #2"
.LC5:
.string "error #1"

```

Listing 21.33: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:

var_10      = -0x10
var_8       = -8
var_4       = -4

        lui      $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x20
        la      $gp, (__gnu_local_gp & 0xFFFF)
        sw      $ra, 0x20+var_4($sp)
        sw      $s0, 0x20+var_8($sp)
        sw      $gp, 0x20+var_10($sp)
        lw      $v0, 0($a0)
        or      $at, $zero

```

```

    slti    $v0, 0x3E9
    bnez    $v0, loc_C8
    move    $s0, $a0
    lw      $v0, 4($a0)
    or      $at, $zero
    sltiu   $v0, 0xB
    bnez    $v0, loc_AC
    lui     $v0, (dword_134 >> 16)
    lwc1    $f4, $LC1
    lwc1    $f2, 8($a0)
    lui     $v0, ($LC2 >> 16)
    lwc1    $f0, 0x14($a0)
    mul.s   $f2, $f4, $f2
    lwc1    $f4, dword_134
    lwc1    $f1, 0x10($a0)
    lwc1    $f5, $LC2
    cvt.d.s $f2, $f2
    mul.d   $f0, $f4, $f0
    lw      $t9, (printf & 0xFFFF)($gp)
    lui     $a0, ($LC0 >> 16) # "%f\n"
    add.d   $f4, $f2, $f0
    mfc1    $a2, $f5
    mfc1    $a3, $f4
    jalr    $t9
    la      $a0, ($LC0 & 0xFFFF) # "%f\n"
    lw      $gp, 0x20+var_10($sp)
    lbu    $a2, 0x19($s0)
    lb      $a1, 0x18($s0)
    lui     $a0, ($LC3 >> 16) # "%c, %d\n"
    lw      $t9, (printf & 0xFFFF)($gp)
    lw      $ra, 0x20+var_4($sp)
    lw      $s0, 0x20+var_8($sp)
    la      $a0, ($LC3 & 0xFFFF) # "%c, %d\n"
    jr      $t9
    addiu   $sp, 0x20
loc_AC:                                # CODE XREF: f+38
    lui     $a0, ($LC4 >> 16) # "error #2"
    lw      $t9, (puts & 0xFFFF)($gp)
    lw      $ra, 0x20+var_4($sp)
    lw      $s0, 0x20+var_8($sp)
    la      $a0, ($LC4 & 0xFFFF) # "error #2"
    jr      $t9
    addiu   $sp, 0x20
loc_C8:                                # CODE XREF: f+24
    lui     $a0, ($LC5 >> 16) # "error #1"
    lw      $t9, (puts & 0xFFFF)($gp)
    lw      $ra, 0x20+var_4($sp)

```

```
lw      $s0, 0x20+var_8($sp)
la      $a0, ($LC5 & 0xFFFF) # "error #1"
jr      $t9
addiu   $sp, 0x20

$LC0:    .ascii "%f\n<0>"
$LC3:    .ascii "%c, %d\n<0>"
$LC4:    .ascii "error #2"<0>
$LC5:    .ascii "error #1"<0>

        .data # .rodata.cst4
$LC1:    .word 0x43DE0000

        .data # .rodata.cst8
$LC2:    .word 0x405EC000
dword_134: .word 0
```

Responda: [G.1.14 on page 1267.](#)

Capítulo 22

22.1

[20 on page 439.](#)

```
#include <stdio.h>
#include <stdint.h>
#include <time.h>

//
//

const uint32_t RNG_a=1664525;
const uint32_t RNG_c=1013904223;
uint32_t RNG_state; //

void my_srand(uint32_t i)
{
    RNG_state=i;
};

uint32_t my_rand()
{
    RNG_state=RNG_state*RNG_a+RNG_c;
    return RNG_state;
};

//

union uint32_t_float
{
    uint32_t i;
    float f;
```

```

};

float float_rand()
{
    union uint32_t_float tmp;
    tmp.i=my_rand() & 0x007fffff | 0x3F800000;
    return tmp.f-1;
};

//


int main()
{
    my_srand(time(NULL)); //

    for (int i=0; i<100; i++)
        printf ("%f\n", float_rand());

    return 0;
};

```

22.1.1 x86

Listing 22.1: Con optimización MSVC 2010

```

$SG4238 DB      '%f', 0AH, 00H

__real@3ff00000000000000 DQ 03ff000000000000r ; 1

tv130 = -4
_tmp$ = -4
?float_rand@@YAMXZ PROC
    push    ecx
    call    ?my_rand@@YAIXZ
; EAX=
    and     eax, 8388607 ; 007✓
    ↳ fffffH
    or      eax, 1065353216 ; 3✓
    ↳ f800000H
; EAX= & 0x007fffff | 0x3f800000
;
    mov     DWORD PTR _tmp$[esp+4], eax
;
    fld     DWORD PTR _tmp$[esp+4]
; 1.0:
    fsub   QWORD PTR __real@3ff00000000000000

```

```
;
    fstp    DWORD PTR tv130[esp+4] ; \
    fld     DWORD PTR tv130[esp+4] ; /
    pop    ecx
    ret    0
?float_rand@@YAMXZ ENDP

_main PROC
    push    esi
    xor     eax, eax
    call    _time
    push    eax
    call    ?my_srand@@YAXI@Z
    add    esp, 4
    mov    esi, 100
$LL3@main:
    call    ?float_rand@@YAMXZ
    sub    esp, 8
    fstp   QWORD PTR [esp]
    push    OFFSET $SG4238
    call    _printf
    add    esp, 12
    dec    esi
    jne    SHORT $LL3@main
    xor    eax, eax
    pop    esi
    ret    0
_main ENDP
```

[51.1.1 on page 702.](#)

[27.5 on page 572.](#)

22.1.2 MIPS

Listing 22.2: Con optimización GCC 4.4.5

```
float_rand:

var_10          = -0x10
var_4           = -4

        lui     $gp, (__gnu_local_gp >> 16)
        addiu  $sp, -0x20
        la     $gp, (__gnu_local_gp & 0xFFFF)
        sw     $ra, 0x20+var_4($sp)
```

```

; my_rand():
    sw      $gp, 0x20+var_10($sp)
    jal     my_rand
    or     $at, $zero ; branch delay slot, NOP
; $v0=
    li      $v1, 0x7FFFFFFF
; $v1=0x7FFFFFFF
    and    $v1, $v0, $v1
; $v1= & 0x7FFFFFFF
    lui     $a0, 0x3F80
; $a0=0x3F800000
    or     $v1, $a0
; $v1= & 0x7FFFFFFF | 0x3F800000
;
    lui     $v0, ($LC0 >> 16)
; $f0:
    lwc1   $f0, $LC0
;
;
    mtc1   $v1, $f2
    lw      $ra, 0x20+var_4($sp)
;
    sub.s  $f0, $f2, $f0
    jr     $ra
    addiu  $sp, 0x20 ; branch delay slot

main:
var_18      = -0x18
var_10      = -0x10
var_C       = -0xC
var_8       = -8
var_4       = -4

    lui     $gp, (__gnu_local_gp >> 16)
    addiu  $sp, -0x28
    la      $gp, (__gnu_local_gp & 0xFFFF)
    sw      $ra, 0x28+var_4($sp)
    sw      $s2, 0x28+var_8($sp)
    sw      $s1, 0x28+var_C($sp)
    sw      $s0, 0x28+var_10($sp)
    sw      $gp, 0x28+var_18($sp)
    lw      $t9, (time & 0xFFFF)($gp)
    or     $at, $zero ; load delay slot, NOP
    jalr   $t9
    move   $a0, $zero ; branch delay slot
    lui     $s2, ($LC1 >> 16) # "%f\n"

```

```

move    $a0, $v0
la      $s2, ($LC1 & 0xFFFF) # "%f\n"
move    $s0, $zero
jal    my_rand
li     $s1, 0x64 # 'd' ; branch delay slot

loc_104:
jal    float_rand
addiu $s0, 1
lw     $gp, 0x28+var_18($sp)
;
cvt.d.s $f2, $f0
lw     $t9, (printf & 0xFFFF)($gp)
mfc1  $a3, $f2
mfc1  $a2, $f3
jalr $t9
move  $a0, $s2
bne   $s0, $s1, loc_104
move  $v0, $zero
lw     $ra, 0x28+var_4($sp)
lw     $s2, 0x28+var_8($sp)
lw     $s1, 0x28+var_C($sp)
lw     $s0, 0x28+var_10($sp)
jr    $ra
addiu $sp, 0x28 ; branch delay slot

$LC1: .ascii "%f\n"<0>
$LC0: .float 1.0

```

[17.5.6 on page 282.](#)

22.1.3 ARM (Modo ARM)

Listing 22.3: Con optimización GCC 4.6.3 (IDA)

```

float_rand
STMFD  SP!, {R3,LR}
BL      my_rand
; R0=
FLDS   S0, =1.0
; S0=1.0
BIC    R3, R0, #0xFF000000
BIC    R3, R3, #0x800000
ORR    R3, R3, #0x3F800000
; R3= & 0x007fffff | 0x3f800000
;

```

```

;
        FMSR    S15, R3
;
        FSUBS   S0, S15, S0
        LDMFD   SP!, {R3,PC}

flt_5C      DCFS 1.0

main
        STMFD   SP!, {R4,LR}
        MOV     R0, #0
        BL      time
        BL      my_srand
        MOV     R4, #0x64 ; 'd'

loc_78
        BL      float_rand
; S0=
        LDR    R0, =aF           ; "%f"
;
        FCVTDS D7, S0
;
        FMRRD   R2, R3, D7
        BL      printf
        SUBS   R4, R4, #1
        BNE    loc_78
        MOV     R0, R4
        LDMFD   SP!, {R4,PC}

aF          DCB  "%f",0xA,0

```

Listing 22.4: Con optimización GCC 4.6.3 (objdump)

00000038 <float_rand>:	
38: e92d4008	push {r3, lr}
3c: ebfffffe	bl 10 <my_rand>
40: ed9f0a05	vldr s0, [pc, #20] ; 5c <2
↳ float_rand+0x24>	
44: e3c034ff	bic r3, r0, #-16777216 ; 0<
↳ xff000000	
48: e3c33502	bic r3, r3, #8388608 ; 0<
↳ x800000	
4c: e38335fe	orr r3, r3, #1065353216 ; 0<
↳ x3f800000	
50: ee073a90	vmov s15, r3
54: ee370ac0	vsub.f32 s0, s15, s0
58: e8bd8008	pop {r3, pc}
5c: 3f800000	svccc 0x00800000

```

00000000 <main>:
    0:   e92d4010      push    {r4, lr}
    4:   e3a00000      mov     r0, #0
    8:   ebfffffe      bl     0 <time>
   c:   ebfffffe      bl     0 <main>
   10:  e3a04064     mov     r4, #100      ; 0x64
   14:  ebfffffe      bl     38 <main+0x38>
   18:  e59f0018      ldr     r0, [pc, #24]    ; 38 <main+0x38>
   ↵ >
   1c:  eeb77ac0      vcvt.f64.f32 d7, s0
   20:  ec532b17      vmov    r2, r3, d7
   24:  ebfffffe      bl     0 <printf>
   28:  e2544001      subs    r4, r4, #1
   2c:  1afffff8      bne    14 <main+0x14>
   30:  e1a00004      mov     r0, r4
   34:  e8bd8010      pop    {r4, pc}
   38:  00000000      andeq   r0, r0, r0

```

22.2

$2^{-23} = 1.19e - 07$ para *float* y $2^{-52} = 2.22e - 16$ para *double*.

```

#include <stdio.h>
#include <stdint.h>

union uint_float
{
    uint32_t i;
    float f;
};

float calculate_machine_epsilon(float start)
{
    union uint_float v;
    v.f=start;
    v.i++;
    return v.f-start;
}

void main()
{
    printf ("%g\n", calculate_machine_epsilon(1.0));
}

```

22.2.1 x86

Listing 22.5: Con optimización MSVC 2010

```

tv130 = 8
_v$ = 8
_start$ = 8
_calculate_machine_epsilon PROC
    fld     DWORD PTR _start$[esp-4]
    fst     DWORD PTR _v$[esp-4]      ;
    inc     DWORD PTR _v$[esp-4]
    fsubr  DWORD PTR _v$[esp-4]
    fstp   DWORD PTR tv130[esp-4]    ; \
    fld     DWORD PTR tv130[esp-4]    ; /
    ret     0
_calculate_machine_epsilon ENDP

```

22.2.2 ARM64

```

#include <stdio.h>
#include <stdint.h>

typedef union
{
    uint64_t i;
    double d;
} uint_double;

double calculate_machine_epsilon(double start)
{
    uint_double v;
    v.d=start;
    v.i++;
    return v.d-start;
}

void main()
{
    printf ("%g\n", calculate_machine_epsilon(1.0));
}

```

Listing 22.6: Con optimización GCC 4.9 ARM64

```

calculate_machine_epsilon:
    fmov    x0, d0          ;

```

```

add      x0, x0, 1      ; X0++
fmov    d1, x0          ;
fsub    d0, d1, d0      ;
ret

```

: 27.4 on page 572.

22.2.3 MIPS

Listing 22.7: Con optimización GCC 4.4.5 (IDA)

```

calculate_machine_epsilon:
    mfc1    $v0, $f12
    or      $at, $zero ; NOP
    addiu   $v1, $v0, 1
    mtc1    $v1, $f2
    jr     $ra
    sub.s   $f0, $f2, $f12 ; branch delay slot

```

22.2.4 Conclusión

22.3

Listing 22.8: <http://go.yurichev.com/17364>

```

/* Assumes that float is in the IEEE 754 single precision ↴
   ↴ floating point format
 * and that int is 32 bits. */
float sqrt_approx(float z)
{
    int val_int = *(int*)&z; /* Same bits, but as an int */
    /*
     * To justify the following code, prove that
     *
     * (((val_int / 2^m) - b) / 2) + b) * 2^m = ((val_int - 2^
     ↴ m) / 2) + ((b + 1) / 2) * 2^m)
     *
     * where
     *
     * b = exponent bias
     * m = number of mantissa bits
     *
     * .
    */

```

```
val_int -= 1 << 23; /* Subtract 2^m. */
val_int >>= 1; /* Divide by 2. */
val_int += 1 << 29; /* Add ((b + 1) / 2) * 2^m. */

return *(float*)&val_int; /* Interpret again as float */
}
```

Como ejercicio, puedes compilar esta función y entender cómo funciona.

$\frac{1}{\sqrt{x}}$.

Wikipedia: .

Capítulo 23

1.

- qsort()², atexit()³;
- ⁴;
- :CreateThread() (win32), pthread_create() (POSIX);
- EnumChildWindows()⁵.
- :<http://go.yurichev.com/17076>
- :<http://go.yurichev.com/17077>
- GitHub.

```
int (*compare)(const void *, const void *)
```

:

```
1 /* ex3 Sorting ints with qsort */
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int comp(const void * _a, const void * _b)
7 {
8     const int *a=(const int *)_a;
9     const int *b=(const int *)_b;
```

¹[wikipedia](#)

²[wikipedia](#)

³<http://go.yurichev.com/17073>

⁴[wikipedia](#)

⁵[MSDN](#)

```

11 if (*a==*b)
12     return 0;
13 else
14     if (*a < *b)
15         return -1;
16     else
17         return 1;
18 }
19
20 int main(int argc, char* argv[])
21 {
22     int numbers ↴
23     ↴ [10]={1892,45,200,-98,4087,5,-12345,1087,88,-100000};
24     int i;
25
26     /* Sort the array */
27     qsort(numbers,10,sizeof(int),comp) ;
28     for (i=0;i<9;i++)
29         printf("Number = %d\n",numbers[ i ]) ;
30     return 0;
}

```

23.1 MSVC

:

Listing 23.1: Con optimización MSVC 2010: /GS- /MD

```

__a$ = 8                                     ; size ↴
    ↴ = 4
__b$ = 12                                    ; size ↴
    ↴ = 4
_comp  PROC
    mov    eax, DWORD PTR __a$[esp-4]
    mov    ecx, DWORD PTR __b$[esp-4]
    mov    eax, DWORD PTR [eax]
    mov    ecx, DWORD PTR [ecx]
    cmp    eax, ecx
    jne    SHORT $LN4@comp
    xor    eax, eax
    ret    0
$LN4@comp:
    xor    edx, edx
    cmp    eax, ecx
    setge dl
    lea    eax, DWORD PTR [edx+edx-1]

```

```

        ret      0
_comp    ENDP

_numbers$ = -40          ; size ↘
    ↴ = 40
_argc$ = 8               ; size ↘
    ↴ = 4
_argv$ = 12              ; size ↘
    ↴ = 4
_main   PROC
    sub     esp, 40          ; ↴
    ↴ 00000028H
        push   esi
        push   OFFSET _comp
        push   4
        lea    eax, DWORD PTR _numbers$[esp+52]
        push   10              ; ↴
    ↴ 000000aH
        push   eax
        mov    DWORD PTR _numbers$[esp+60], 1892      ; ↴
    ↴ 00000764H
        mov    DWORD PTR _numbers$[esp+64], 45          ; ↴
    ↴ 0000002dH
        mov    DWORD PTR _numbers$[esp+68], 200         ; ↴
    ↴ 000000c8H
        mov    DWORD PTR _numbers$[esp+72], -98         ; ↴
    ↴ ffffff9eH
        mov    DWORD PTR _numbers$[esp+76], 4087        ; 00000000
    ↴ ff7H
        mov    DWORD PTR _numbers$[esp+80], 5
        mov    DWORD PTR _numbers$[esp+84], -12345       ; ↴
    ↴ fffffcfc7H
        mov    DWORD PTR _numbers$[esp+88], 1087         ; ↴
    ↴ 0000043fH
        mov    DWORD PTR _numbers$[esp+92], 88          ; ↴
    ↴ 00000058H
        mov    DWORD PTR _numbers$[esp+96], -100000       ; ↴
    ↴ fffe7960H
        call   _qsort
        add    esp, 16          ; ↴
    ↴ 00000010H
...

```

Listing 23.2: MSVCR80.DLL

```
.text:7816CBF0 ; void __cdecl qsort(void *, unsigned int, ↴
    ↴ unsigned int, int (__cdecl *)(const void *, const void *)) ↴
    ↴ )
.text:7816CBF0                 public _qsort
.text:7816CBF0 _qsort          proc near
.text:7816CBF0
.text:7816CBF0 lo             = dword ptr -104h
.text:7816CBF0 hi             = dword ptr -100h
.text:7816CBF0 var_FC         = dword ptr -0FCh
.text:7816CBF0 stkptr          = dword ptr -0F8h
.text:7816CBF0 lostk           = dword ptr -0F4h
.text:7816CBF0 histk           = dword ptr -7Ch
.text:7816CBF0 base            = dword ptr 4
.text:7816CBF0 num             = dword ptr 8
.text:7816CBF0 width           = dword ptr 0Ch
.text:7816CBF0 comp            = dword ptr 10h
.text:7816CBF0
.text:7816CBF0                 sub     esp, 100h

....
```

.

```
.text:7816CCE0 loc_7816CCE0:           ; CODE ↴
    ↴ XREF: _qsort+B1
.text:7816CCE0 shr    eax, 1
.text:7816CCE2 imul   eax, ebp
.text:7816CCE5 add    eax, ebx
.text:7816CCE7 mov    edi, eax
.text:7816CCE9 push   edi
.text:7816CCEA push   ebx
.text:7816CCEB call   [esp+118h+comp]
.text:7816CCF2 add    esp, 8
.text:7816CCF5 test   eax, eax
.text:7816CCF7 jle    short loc_7816CD04
```

comp

23.1.1 MSVC + OllyDbg

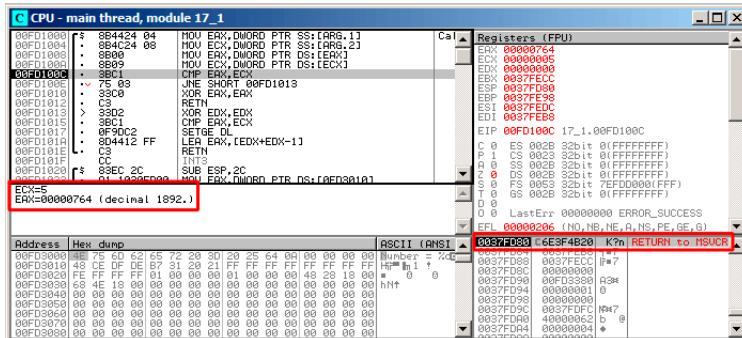


Figura 23.1: OllyDbg: `comp()`

OllyDbg . SP RA `qsort()` (MSVCR100.DLL).

(F8) RETN qsort():

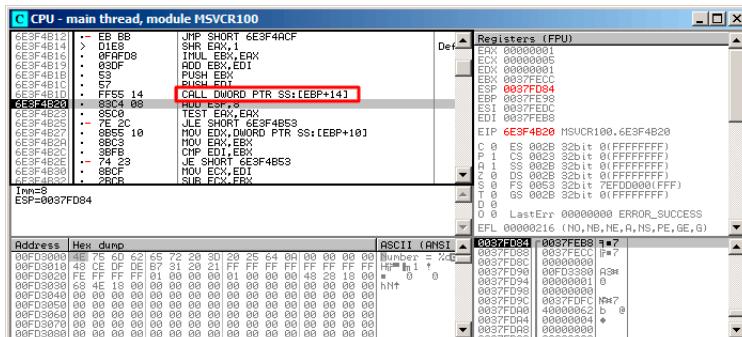


Figura 23.2: OllyDbg: qsort() comp()

comp() :

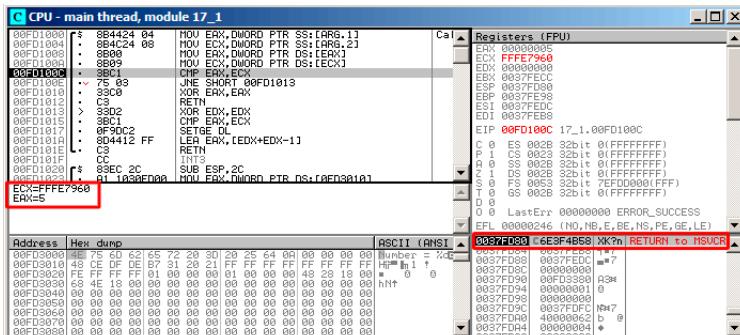


Figura 23.3: OllyDbg: comp()

23.1.2 MSVC + tracer

.: 1892, 45, 200, -98, 4087, 5, -12345, 1087, 88, -100000.

```
tracer.exe -l:17_1.exe bpx=17_1.exe!0x0040100c
```

```
PID=4336|New process 17_1.exe
(0) 17_1.exe!0x40100c
EAX=0x00000764 EBX=0x0051f7c8 ECX=0x00000005 EDX=0x00000000
ESI=0x0051f7d8 EDI=0x0051f7b4 EBP=0x0051f794 ESP=0x0051f67c
EIP=0x0028100c
FLAGS=IF
(0) 17_1.exe!0x40100c
EAX=0x00000005 EBX=0x0051f7c8 ECX=0xffffe7960 EDX=0x00000000
ESI=0x0051f7d8 EDI=0x0051f7b4 EBP=0x0051f794 ESP=0x0051f67c
EIP=0x0028100c
FLAGS=PF ZF IF
(0) 17_1.exe!0x40100c
EAX=0x00000764 EBX=0x0051f7c8 ECX=0x00000005 EDX=0x00000000
ESI=0x0051f7d8 EDI=0x0051f7b4 EBP=0x0051f794 ESP=0x0051f67c
EIP=0x0028100c
FLAGS=CF PF ZF IF
...
```

```
EAX=0x00000764 ECX=0x00000005
EAX=0x00000005 ECX=0xffffe7960
EAX=0x00000764 ECX=0x00000005
EAX=0x0000002d ECX=0x00000005
```

```
EAX=0x00000058 ECX=0x00000005
EAX=0x0000043f ECX=0x00000005
EAX=0xfffffcfc7 ECX=0x00000005
EAX=0x000000c8 ECX=0x00000005
EAX=0xfffffff9e ECX=0x00000005
EAX=0x00000ff7 ECX=0x00000005
EAX=0x00000ff7 ECX=0x00000005
EAX=0xfffffff9e ECX=0x00000005
EAX=0xfffffff9e ECX=0x00000005
EAX=0xfffffcfc7 ECX=0xffffe7960
EAX=0x00000005 ECX=0xfffffcfc7
EAX=0xfffffff9e ECX=0x00000005
EAX=0xfffffcfc7 ECX=0xffffe7960
EAX=0xfffffff9e ECX=0xfffffcfc7
EAX=0xfffffcfc7 ECX=0xffffe7960
EAX=0x000000058 ECX=0x00000ff7
EAX=0x00000002d ECX=0x00000ff7
EAX=0x0000043f ECX=0x00000ff7
EAX=0x00000058 ECX=0x00000ff7
EAX=0x000000764 ECX=0x00000ff7
EAX=0x000000c8 ECX=0x00000764
EAX=0x00000002d ECX=0x00000764
EAX=0x0000043f ECX=0x00000764
EAX=0x00000058 ECX=0x00000764
EAX=0x000000c8 ECX=0x00000058
EAX=0x00000002d ECX=0x000000c8
EAX=0x0000043f ECX=0x000000c8
EAX=0x000000c8 ECX=0x00000058
EAX=0x00000002d ECX=0x000000c8
EAX=0x00000002d ECX=0x00000058
```

23.1.3 MSVC + tracer (code coverage)

```
tracer.exe -l:17_1.exe bpf=17_1.exe!0x00401000,trace:cc
```

```
.text:00401000
.text:00401000 ; int __cdecl PtFuncCompare(const void *, const void *)
.text:00401000 PtFuncCompare    proc near             ; DATA XREF: _main+5j
.text:00401000
.text:00401000     = dword ptr  4
.text:00401000     = dword ptr  8
.text:00401000
.text:00401000     mov    eax, [esp+arg_0] ; [ESP+4]=0x45F7ec..0x45F810(step=4), L"?\\x04?
.text:00401004     mov    ecx, [esp+arg_4] ; [ESP+8]=0x45F7ec..0x45F7F4(step=4), 0x45F7Fc
.text:00401008     mov    eax, [eax]      ; [EAX]=5, 0x2d, 0x58, 0xc8, 0x43f, 0x764, 0xFF
.text:0040100c     mov    ecx, [ecx]      ; [ECX]=5, 0x58, 0xc8, 0x764, 0xFF7, 0xFFFFe7960
.text:0040100e     cmp    eax, ecx       ; EAX=5, 0x2d, 0x58, 0xc8, 0x43f, 0x764, 0xFF7,
.text:00401010     jnz    short loc_401013 ; ZF=False
.text:00401012     xor    eax, eax
.text:00401013     retn
.text:00401013 ; -----
.text:00401013 loc_401013:           ; CODE XREF: PtFuncCompare+E7j
.text:00401013     xor    edx, edx
.text:00401015     cmp    eax, ecx       ; EAX=5, 0x2d, 0x58, 0xc8, 0x43f, 0x764, 0xFF7,
.text:00401017     setnl dl          ; SF=False,true OF=False
.text:0040101a     lea    eax, [edx+edx-1]
.text:0040101e     retn
.text:0040101f PtFuncCompare    endp
.text:0040101f
```

Figura 23.4: tracer y IDA. N.B.:

0x401010 y 0x401012 (): comp().

23.2 GCC

Listing 23.3: GCC

```
lea    eax, [esp+40h+var_28]
mov   [esp+40h+var_40], eax
mov   [esp+40h+var_28], 764h
mov   [esp+40h+var_24], 2Dh
mov   [esp+40h+var_20], 0C8h
mov   [esp+40h+var_1C], 0FFFFFF9Eh
mov   [esp+40h+var_18], 0FF7h
mov   [esp+40h+var_14], 5
```

```

    mov      [esp+40h+var_10], 0FFFFFCFC7h
    mov      [esp+40h+var_C], 43Fh
    mov      [esp+40h+var_8], 58h
    mov      [esp+40h+var_4], 0FFE7960h
    mov      [esp+40h+var_34], offset comp
    mov      [esp+40h+var_38], 4
    mov      [esp+40h+var_3C], 0Ah
    call     _qsort

```

:

```

comp          public comp
comp          proc near

arg_0         = dword ptr  8
arg_4         = dword ptr  0Ch

push    ebp
mov     ebp, esp
mov     eax, [ebp+arg_4]
mov     ecx, [ebp+arg_0]
mov     edx, [eax]
xor     eax, eax
cmp     [ecx], edx
jnz     short loc_8048458
pop     ebp
retn

loc_8048458:
setnl  al
movzx  eax, al
lea    eax, [eax+eax-1]
pop    ebp
retn

comp        endp

```

Listing 23.4: (2.10.1)

.text:0002DDF6	mov edx, [ebp+arg_10]
.text:0002DDF9	mov [esp+4], esi
.text:0002DDFD	mov [esp], edi
.text:0002DE00	mov [esp+8], edx
.text:0002DE04	call [ebp+arg_C]
...	

23.2.1 GCC + GDB ()

```
(0.. (p):.
(bt) msort_with_tmp().
```

Listing 23.5: GDB

```
dennis@ubuntuvm:~/polygon$ gcc 17_1.c -g
dennis@ubuntuvm:~/polygon$ gdb ./a.out
GNU gdb (GDB) 7.6.1-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/遵守
    ↴ licenses/gpl.html>
This is free software: you are free to change and redistribute ↴
    ↴ it.
There is NO WARRANTY, to the extent permitted by law. Type "↙
    ↴ show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/dennis/polygon/a.out...done.
(gdb) b 17_1.c:11
Breakpoint 1 at 0x804845f: file 17_1.c, line 11.
(gdb) run
Starting program: /home/dennis/polygon./a.out

Breakpoint 1, comp (_a=0xbffff0f8, _b=_b@entry=0xbffff0fc) at ↴
    ↴ 17_1.c:11
11      if (*a==*b)
(gdb) p *a
$1 = 1892
(gdb) p *b
$2 = 45
(gdb) c
Continuing.

Breakpoint 1, comp (_a=0xbffff104, _b=_b@entry=0xbffff108) at ↴
    ↴ 17_1.c:11
11      if (*a==*b)
(gdb) p *a
$3 = -98
(gdb) p *b
$4 = 4087
(gdb) bt
#0  comp (_a=0xbffff0f8, _b=_b@entry=0xbffff0fc) at 17_1.c:11
#1  0xb7e42872 in msort_with_tmp (p=p@entry=0xbffff07c, b=↙
    ↴ b@entry=0xbffff0f8, n=n@entry=2)
```

```

at msort.c:65
#2 0xb7e4273e in msort_with_tmp (n=2, b=0xbffff0f8, p=0x
 ↴ xffff07c) at msort.c:45
#3 msort_with_tmp (p=p@entry=0xbffff07c, b=b@entry=0xbffff0f8, ↴
 ↴ n=n@entry=5) at msort.c:53
#4 0xb7e4273e in msort_with_tmp (n=5, b=0xbffff0f8, p=0x
 ↴ xffff07c) at msort.c:45
#5 msort_with_tmp (p=p@entry=0xbffff07c, b=b@entry=0xbffff0f8, ↴
 ↴ n=n@entry=10) at msort.c:53
#6 0xb7e42cef in msort_with_tmp (n=10, b=0xbffff0f8, p=0x
 ↴ xffff07c) at msort.c:45
#7 __GI_qsort_r (b=b@entry=0xbffff0f8, n=n@entry=10, s=s@entry
 ↴ =4, cmp=cmp@entry=0x804844d <comp>,
 arg=arg@entry=0x0) at msort.c:297
#8 0xb7e42dcf in __GI_qsort (b=0xbffff0f8, n=10, s=4, cmp=0x
 ↴ x804844d <comp>) at msort.c:307
#9 0x0804850d in main (argc=1, argv=0xbffff1c4) at 17_1.c:26
(gdb)

```

23.2.2 GCC + GDB ()

(disas), (b). (info registers). (bt), .

Listing 23.6: GDB

```

dennis@ubuntuvm:~/polygon$ gcc 17_1.c
dennis@ubuntuvm:~/polygon$ gdb ./a.out
GNU gdb (GDB) 7.6.1-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/dennis/polygon/a.out...(no debugging
 ↴ symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb) disas comp
Dump of assembler code for function comp:
0x0804844d <+0>:    push    ebp
0x0804844e <+1>:    mov     ebp,esp
0x08048450 <+3>:    sub     esp,0x10

```

0x08048453 <+6>:	mov	eax,DWORD PTR [ebp+0x8]
0x08048456 <+9>:	mov	DWORD PTR [ebp-0x8],eax
0x08048459 <+12>:	mov	eax,DWORD PTR [ebp+0xc]
0x0804845c <+15>:	mov	DWORD PTR [ebp-0x4],eax
0x0804845f <+18>:	mov	eax,DWORD PTR [ebp-0x8]
0x08048462 <+21>:	mov	edx,DWORD PTR [eax]
0x08048464 <+23>:	mov	eax,DWORD PTR [ebp-0x4]
0x08048467 <+26>:	mov	eax,DWORD PTR [eax]
0x08048469 <+28>:	cmp	edx, eax
0x0804846b <+30>:	jne	0x8048474 <comp+39>
0x0804846d <+32>:	mov	eax,0x0
0x08048472 <+37>:	jmp	0x804848e <comp+65>
0x08048474 <+39>:	mov	eax,DWORD PTR [ebp-0x8]
0x08048477 <+42>:	mov	edx,DWORD PTR [eax]
0x08048479 <+44>:	mov	eax,DWORD PTR [ebp-0x4]
0x0804847c <+47>:	mov	eax,DWORD PTR [eax]
0x0804847e <+49>:	cmp	edx, eax
0x08048480 <+51>:	jge	0x8048489 <comp+60>
0x08048482 <+53>:	mov	eax,0xffffffff
0x08048487 <+58>:	jmp	0x804848e <comp+65>
0x08048489 <+60>:	mov	eax,0x1
0x0804848e <+65>:	leave	
0x0804848f <+66>:	ret	

End of assembler dump.

(gdb) b *0x08048469

Breakpoint 1 at 0x8048469

(gdb) run

Starting program: /home/dennis/polygon/.a.out

Breakpoint 1, 0x08048469 in comp ()

(gdb) info registers

eax	0x2d	45
ecx	0xbfffff0f8	-1073745672
edx	0x764	1892
ebx	0xb7fc0000	-1208221696
esp	0xbffffeeb8	0xbffffeeb8
ebp	0xbffffeec8	0xbffffeec8
esi	0xbfffff0fc	-1073745668
edi	0xbfffff010	-1073745904
eip	0x8048469	0x8048469 <comp+28>
eflags	0x286	[PF SF IF]
cs	0x73	115
ss	0x7b	123
ds	0x7b	123
es	0x7b	123
fs	0x0	0
gs	0x33	51

```
(gdb) c
Continuing.
```

```
Breakpoint 1, 0x08048469 in comp ()
(gdb) info registers
eax          0xffff7    4087
ecx          0xbfffff104   -1073745660
edx          0xffffffff9e    -98
ebx          0xb7fc0000   -1208221696
esp          0xbfffffee58   0xbfffffee58
ebp          0xbfffffee68   0xbfffffee68
esi          0xbfffff108   -1073745656
edi          0xbfffff010   -1073745904
eip          0x8048469    0x8048469 <comp+28>
eflags        0x282      [ SF IF ]
cs            0x73       115
ss            0x7b       123
ds            0x7b       123
es            0x7b       123
fs            0x0        0
gs            0x33       51
```

```
(gdb) c
Continuing.
```

```
Breakpoint 1, 0x08048469 in comp ()
(gdb) info registers
eax          0xffffffff9e    -98
ecx          0xbfffff100   -1073745664
edx          0xc8        200
ebx          0xb7fc0000   -1208221696
esp          0xbffffeeb8   0xbffffeeb8
ebp          0xbffffeec8   0xbffffeec8
esi          0xbfffff104   -1073745660
edi          0xbfffff010   -1073745904
eip          0x8048469    0x8048469 <comp+28>
eflags        0x286      [ PF SF IF ]
cs            0x73       115
ss            0x7b       123
ds            0x7b       123
es            0x7b       123
fs            0x0        0
gs            0x33       51
```

```
(gdb) bt
```

```
#0 0x08048469 in comp ()
#1 0xb7e42872 in msort_with_tmp (p=p@entry=0xbfffff07c, b=b@entry=0xbfffff0f8, n=n@entry=2)
    ↳ b@entry=0xbfffff0f8, n=n@entry=2
    at msort.c:65
```

```
#2 0xb7e4273e in msort_with_tmp (n=2, b=0xbffff0f8, p=0x2
 ↴ xffff07c) at msort.c:45
#3 msort_with_tmp (p=p@entry=0xbffff07c, b=b@entry=0xbffff0f8, ↴
 ↴ n=n@entry=5) at msort.c:53
#4 0xb7e4273e in msort_with_tmp (n=5, b=0xbffff0f8, p=0x2
 ↴ xffff07c) at msort.c:45
#5 msort_with_tmp (p=p@entry=0xbffff07c, b=b@entry=0xbffff0f8, ↴
 ↴ n=n@entry=10) at msort.c:53
#6 0xb7e42cef in msort_with_tmp (n=10, b=0xbffff0f8, p=0x2
 ↴ xffff07c) at msort.c:45
#7 __GI_qsort_r (b=b@entry=0xbffff0f8, n=n@entry=10, s=s@entry
 ↴ =4, cmp=cmp@entry=0x804844d <comp>,
 arg=arg@entry=0x0) at msort.c:297
#8 0xb7e42dcf in __GI_qsort (b=0xbffff0f8, n=10, s=4, cmp=0x2
 ↴ x804844d <comp>) at msort.c:307
#9 0x0804850d in main ()
```

Capítulo 24

¹.

24.1

```
#include <stdint.h>

uint64_t f ()
{
    return 0x1234567890ABCDEF;
}
```

24.1.1 x86

Listing 24.1: Con optimización MSVC 2010

```
_f      PROC
        mov     eax, -1867788817          ; 90abcdefH
        mov     edx, 305419896           ; 12345678H
        ret     0
_f      ENDP
```

24.1.2 ARM

Listing 24.2: Con optimización Keil 6/2013 (Modo ARM)

```
||f|| PROC
        LDR     r0, |L0.12|
        LDR     r1, |L0.16|
```

¹: [53.4 on page 779](#)

```

BX      lr
ENDP

|L0.12|
    DCD    0x90abcdef
|L0.16|
    DCD    0x12345678

```

24.1.3 MIPS

Listing 24.3: Con optimización GCC 4.4.5 (assembly listing)

```

li      $3,-1867841536          # 0x
↳ xfffffff90ab0000
    li      $2,305397760         # 0x12340000
    ori    $3,$3,0xcdef
    j       $31
    ori    $2,$2,0x5678

```

Listing 24.4: Con optimización GCC 4.4.5 (IDA)

```

lui    $v1, 0x90AB
lui    $v0, 0x1234
li     $v1, 0x90ABCDEF
jr     $ra
li     $v0, 0x12345678

```

24.2

```

#include <stdint.h>

uint64_t f_add (uint64_t a, uint64_t b)
{
    return a+b;
};

void f_add_test ()
{
#ifdef __GNUC__
    printf ("%lld\n", f_add(12345678901234, 2345678901234));
#endif
    else
        printf ("%I64d\n", f_add(12345678901234,
                                2345678901234));
}

```

```
#endif
};

uint64_t f_sub (uint64_t a, uint64_t b)
{
    return a-b;
};
```

24.2.1 x86

Listing 24.5: Con optimización MSVC 2012 /Ob1

```
_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_f_add PROC
    mov     eax, DWORD PTR _a$[esp-4]
    add     eax, DWORD PTR _b$[esp-4]
    mov     edx, DWORD PTR _a$[esp]
    adc     edx, DWORD PTR _b$[esp]
    ret     0
_f_add ENDP

_f_add_test PROC
    push    5461           ; 00001555H
    push    1972608889    ; 75939f79H
    push    2874            ; 00000b3aH
    push    1942892530    ; 73ce2ff_subH
    call    _f_add
    push    edx
    push    eax
    push    OFFSET $SG1436 ; '%I64d', 0aH, 00H
    call    _printf
    add    esp, 28
    ret     0
_f_add_test ENDP

_f_sub PROC
    mov     eax, DWORD PTR _a$[esp-4]
    sub     eax, DWORD PTR _b$[esp-4]
    mov     edx, DWORD PTR _a$[esp]
    sbb     edx, DWORD PTR _b$[esp]
    ret     0
_f_sub ENDP
```

f_add_test().

. SUB : .

Listing 24.6: GCC 4.8.1 -O1 -fno-inline

```
_f_add:
    mov     eax, DWORD PTR [esp+12]
    mov     edx, DWORD PTR [esp+16]
    add     eax, DWORD PTR [esp+4]
    adc     edx, DWORD PTR [esp+8]
    ret

_f_add_test:
    sub    esp, 28
    mov    DWORD PTR [esp+8], 1972608889 ; 75939f79H
    mov    DWORD PTR [esp+12], 5461      ; 00001555H
    mov    DWORD PTR [esp], 1942892530 ; 73ce2ff_subH
    mov    DWORD PTR [esp+4], 2874      ; 00000b3aH
    call   _f_add
    mov    DWORD PTR [esp+4], eax
    mov    DWORD PTR [esp+8], edx
    mov    DWORD PTR [esp], OFFSET FLAT:LC0 ; "%lld\12\0"
    call   _printf
    add    esp, 28
    ret

_f_sub:
    mov     eax, DWORD PTR [esp+4]
    mov     edx, DWORD PTR [esp+8]
    sub     eax, DWORD PTR [esp+12]
    sbb     edx, DWORD PTR [esp+16]
    ret
```

24.2.2 ARM

Listing 24.7: Con optimización Keil 6/2013 (Modo ARM)

```
f_add PROC
    ADDS    r0,r0,r2
    ADC     r1,r1,r3
```

```

        BX      lr
        ENDP

f_sub PROC
        SUBS   r0,r0,r2
        SBC    r1,r1,r3
        BX     lr
        ENDP

f_add_test PROC
        PUSH   {r4,lr}
        LDR    r2,|L0.68| ; 0x75939f79
        LDR    r3,|L0.72| ; 0x00001555
        LDR    r0,|L0.76| ; 0x73ce2ff2
        LDR    r1,|L0.80| ; 0x00000b3a
        BL    f_add
        POP    {r4,lr}
        MOV    r2,r0
        MOV    r3,r1
        ADR    r0,|L0.84| ; "%I64d\n"
        B     __2printf
        ENDP

|L0.68|
        DCD    0x75939f79
|L0.72|
        DCD    0x00001555
|L0.76|
        DCD    0x73ce2ff2
|L0.80|
        DCD    0x00000b3a
|L0.84|
        DCB    "%I64d\n",0

```

24.2.3 MIPS

Listing 24.8: Con optimización GCC 4.4.5 (IDA)

```

f_add:
; $a0 - a
; $a1 - a
; $a2 - b
; $a3 - b
            addu   $v1, $a3, $a1 ;
            addu   $a0, $a2, $a0 ;
;

```

```
;
        sltu    $v0, $v1, $a3
        jr     $ra
;
        addu    $v0, $a0 ; branch delay slot
; $v0 -
; $v1 -

f_sub:
; $a0 - a
; $a1 - a
; $a2 - b
; $a3 - b
        subu    $v1, $a1, $a3 ;
        subu    $v0, $a0, $a2 ;
;
;
        sltu    $a1, $v1
        jr     $ra
;
        subu    $v0, $a1 ; branch delay slot
; $v0 -
; $v1 -

f_add_test:

var_10      = -0x10
var_4       = -4

        lui     $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x20
        la      $gp, (__gnu_local_gp & 0xFFFF)
        sw      $ra, 0x20+var_4($sp)
        sw      $gp, 0x20+var_10($sp)
        lui     $a1, 0x73CE
        lui     $a3, 0x7593
        li      $a0, 0xB3A
        li      $a3, 0x75939F79
        li      $a2, 0x1555
        jal    f_add
        li      $a1, 0x73CE2FF2
        lw      $gp, 0x20+var_10($sp)
        lui     $a0, ($LC0 >> 16) # "%lld\n"
        lw      $t9, (printf & 0xFFFF)($gp)
        lw      $ra, 0x20+var_4($sp)
        la      $a0, ($LC0 & 0xFFFF) # "%lld\n"
        move   $a3, $v1
```

```

        move    $a2, $v0
        jr     $t9
        addiu   $sp, 0x20

$LCO:      .ascii "%lld\n"<0>

```

24.3

```

#include <stdint.h>

uint64_t f_mul (uint64_t a, uint64_t b)
{
    return a*b;
};

uint64_t f_div (uint64_t a, uint64_t b)
{
    return a/b;
};

uint64_t f_rem (uint64_t a, uint64_t b)
{
    return a % b;
};

```

24.3.1 x86

Listing 24.9: Con optimización MSVC 2013 /Ob1

```

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_mul PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __allmul ; long long multiplication
    pop     ebp

```

```

        ret      0
_f_mul ENDP

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_div PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __aulldiv ; unsigned long long division
    pop     ebp
    ret      0
_f_div ENDP

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_rem PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __aullrem ; unsigned long long remainder
    pop     ebp
    ret      0
_f_rem ENDP

```

: E on page 1254.

Listing 24.10: Con optimización GCC 4.8.1 -fno-inline

```

_f_mul:
    push    ebx

```

```

    mov    edx, DWORD PTR [esp+8]
    mov    eax, DWORD PTR [esp+16]
    mov    ebx, DWORD PTR [esp+12]
    mov    ecx, DWORD PTR [esp+20]
    imul   ebx, eax
    imul   ecx, edx
    mul    edx
    add    ecx, ebx
    add    edx, ecx
    pop    ebx
    ret

_f_div:
    sub    esp, 28
    mov    eax, DWORD PTR [esp+40]
    mov    edx, DWORD PTR [esp+44]
    mov    DWORD PTR [esp+8], eax
    mov    eax, DWORD PTR [esp+32]
    mov    DWORD PTR [esp+12], edx
    mov    edx, DWORD PTR [esp+36]
    mov    DWORD PTR [esp], eax
    mov    DWORD PTR [esp+4], edx
    call   __udivdi3 ; unsigned division
    add    esp, 28
    ret

_f_rem:
    sub    esp, 28
    mov    eax, DWORD PTR [esp+40]
    mov    edx, DWORD PTR [esp+44]
    mov    DWORD PTR [esp+8], eax
    mov    eax, DWORD PTR [esp+32]
    mov    DWORD PTR [esp+12], edx
    mov    edx, DWORD PTR [esp+36]
    mov    DWORD PTR [esp], eax
    mov    DWORD PTR [esp+4], edx
    call   __umoddi3 ; unsigned modulo
    add    esp, 28
    ret

```

[.: D on page 1253.](#)

24.3.2 ARM

Listing 24.11: Con optimización Keil 6/2013 (Modo Thumb)

f_mul PROC

```

PUSH    {r4,lr}
BL      __aeabi_lmul
POP    {r4,pc}
ENDP

||f_div|| PROC
PUSH    {r4,lr}
BL      __aeabi_ulddivmod
POP    {r4,pc}
ENDP

||f_rem|| PROC
PUSH    {r4,lr}
BL      __aeabi_ulddivmod
MOVS   r0,r2
MOVS   r1,r3
POP    {r4,pc}
ENDP

```

Listing 24.12: Con optimización Keil 6/2013 (Modo ARM)

```

||f_mul|| PROC
PUSH    {r4,lr}
UMULL  r12,r4,r0,r2
MLA    r1,r2,r1,r4
MLA    r1,r0,r3,r1
MOV    r0,r12
POP    {r4,pc}
ENDP

||f_div|| PROC
PUSH    {r4,lr}
BL      __aeabi_ulddivmod
POP    {r4,pc}
ENDP

||f_rem|| PROC
PUSH    {r4,lr}
BL      __aeabi_ulddivmod
MOV    r0,r2
MOV    r1,r3
POP    {r4,pc}
ENDP

```

24.3.3 MIPS

Con optimización GCC para MIPS

Listing 24.13: Con optimización GCC 4.4.5 (IDA)

```

f_mul:
    mult    $a2, $a1
    mflo    $v0
    or      $at, $zero ; NOP
    or      $at, $zero ; NOP
    mult    $a0, $a3
    mflo    $a0
    addu   $v0, $a0
    or      $at, $zero ; NOP
    multu   $a3, $a1
    mfhi   $a2
    mflo   $v1
    jr     $ra
    addu   $v0, $a2

f_div:
var_10      = -0x10
var_4       = -4

    lui     $gp, (__gnu_local_gp >> 16)
    addiu  $sp, -0x20
    la     $gp, (__gnu_local_gp & 0xFFFF)
    sw     $ra, 0x20+var_4($sp)
    sw     $gp, 0x20+var_10($sp)
    lw     $t9, (__udivdi3 & 0xFFFF)($gp)
    or     $at, $zero
    jalr   $t9
    or     $at, $zero
    lw     $ra, 0x20+var_4($sp)
    or     $at, $zero
    jr     $ra
    addiu $sp, 0x20

f_rem:
var_10      = -0x10
var_4       = -4

    lui     $gp, (__gnu_local_gp >> 16)
    addiu  $sp, -0x20
    la     $gp, (__gnu_local_gp & 0xFFFF)
    sw     $ra, 0x20+var_4($sp)
    sw     $gp, 0x20+var_10($sp)
    lw     $t9, (__umoddi3 & 0xFFFF)($gp)
    or     $at, $zero

```

```

jalr    $t9
or     $at, $zero
lw     $ra, 0x20+var_4($sp)
or     $at, $zero
jr     $ra
addiu $sp, 0x20

```

24.4

```

#include <stdint.h>

uint64_t f (uint64_t a)
{
    return a>>7;
}

```

24.4.1 x86

Listing 24.14: Con optimización MSVC 2012 /O_{b1}

```

_a$ = 8          ; size = 8
_f      PROC
    mov    eax, DWORD PTR _a$[esp-4]
    mov    edx, DWORD PTR _a$[esp]
    shrd   eax, edx, 7
    shr    edx, 7
    ret    0
_f      ENDP

```

Listing 24.15: Con optimización GCC 4.8.1 -fno-inline

```

_f:
    mov    edx, DWORD PTR [esp+8]
    mov    eax, DWORD PTR [esp+4]
    shrd   eax, edx, 7
    shr    edx, 7
    ret

```

24.4.2 ARM

Listing 24.16: Con optimización Keil 6/2013 (Modo ARM)

```
||f|| PROC
    LSR      r0,r0,#7
    ORR      r0,r0,r1,LSL #25
    LSR      r1,r1,#7
    BX       lr
ENDP
```

Listing 24.17: Con optimización Keil 6/2013 (Modo Thumb)

```
||f|| PROC
    LSLS     r2,r1,#25
    LSRS     r0,r0,#7
    ORRS     r0,r0,r2
    LSRS     r1,r1,#7
    BX       lr
ENDP
```

24.4.3 MIPS

Listing 24.18: Con optimización GCC 4.4.5 (IDA)

```
f:
    sll      $v0, $a0, 25
    srl      $v1, $a1, 7
    or      $v1, $v0, $v1
    jr      $ra
    srl      $v0, $a0, 7
```

24.5

```
#include <stdint.h>

int64_t f (int32_t a)
{
    return a;
}
```

24.5.1 x86

Listing 24.19: Con optimización MSVC 2012

```
_a$ = 8
```

```
_f      PROC
        mov      eax, DWORD PTR _a$[esp-4]
        cdq
        ret      0
_f      ENDP
```

· · · · ·

24.5.2 ARM

Listing 24.20: Con optimización Keil 6/2013 (Modo ARM)

```
||f|| PROC
        ASR      r1,r0,#31
        BX       lr
ENDP
```

24.5.3 MIPS

Listing 24.21: Con optimización GCC 4.4.5 (IDA)

```
f:
        sra      $v0, $a0, 31
        jr      $ra
        move   $v1, $a0
```

Capítulo 25

SIMD

SIMD¹ : Single Instruction, Multiple Data.

SSE

AVX

<http://go.yurichev.com/17313>

25.1

```
.  
for (i = 0; i < 1024; i++)  
{  
    C[i] = A[i]*B[i];  
}
```

25.1.1

```
int f (int sz, int *ar1, int *ar2, int *ar3)  
{  
    for (int i=0; i<sz; i++)  
        ar3[i]=ar1[i]+ar2[i];  
  
    return 0;  
};
```

¹Single instruction, multiple data

Intel C++

Intel C++ 11.1.051 win32:

icl intel.cpp /QaxSSE2 /Faintel.asm /Ox

```
; int __cdecl f(int, int *, int *, int *)
    public ?f@@YAHHPAH00@Z
?f@@YAHHPAH00@Z proc near

var_10 = dword ptr -10h
sz      = dword ptr 4
ar1     = dword ptr 8
ar2     = dword ptr 0Ch
ar3     = dword ptr 10h

    push    edi
    push    esi
    push    ebx
    push    esi
    mov     edx, [esp+10h+sz]
    test   edx, edx
    jle    loc_15B
    mov     eax, [esp+10h+ar3]
    cmp    edx, 6
    jle    loc_143
    cmp    eax, [esp+10h+ar2]
    jbe    short loc_36
    mov     esi, [esp+10h+ar2]
    sub    esi, eax
    lea    ecx, ds:0[edx*4]
    neg    esi
    cmp    ecx, esi
    jbe    short loc_55

loc_36: ; CODE XREF: f(int,int *,int *,int *)+21
    cmp    eax, [esp+10h+ar2]
    jnb    loc_143
    mov    esi, [esp+10h+ar2]
    sub    esi, eax
    lea    ecx, ds:0[edx*4]
    cmp    esi, ecx
    jb     loc_143

loc_55: ; CODE XREF: f(int,int *,int *,int *)+34
    cmp    eax, [esp+10h+ar1]
    jbe    short loc_67
```

```
    mov    esi, [esp+10h+ar1]
    sub    esi, eax
    neg    esi
    cmp    ecx, esi
    jbe    short loc_7F

loc_67: ; CODE XREF: f(int,int *,int *,int *)+59
    cmp    eax, [esp+10h+ar1]
    jnb    loc_143
    mov    esi, [esp+10h+ar1]
    sub    esi, eax
    cmp    esi, ecx
    jb     loc_143

loc_7F: ; CODE XREF: f(int,int *,int *,int *)+65
    mov    edi, eax      ; edi = ar1
    and    edi, 0Fh      ; ?
    jz    short loc_9A   ;
    test   edi, 3
    jnz    loc_162
    neg    edi
    add    edi, 10h
    shr    edi, 2

loc_9A: ; CODE XREF: f(int,int *,int *,int *)+84
    lea    ecx, [edi+4]
    cmp    edx, ecx
    jl    loc_162
    mov    ecx, edx
    sub    ecx, edi
    and    ecx, 3
    neg    ecx
    add    ecx, edx
    test   edi, edi
    jbe    short loc_D6
    mov    ebx, [esp+10h+ar2]
    mov    [esp+10h+var_10], ecx
    mov    ecx, [esp+10h+ar1]
    xor    esi, esi

loc_C1: ; CODE XREF: f(int,int *,int *,int *)+CD
    mov    edx, [ecx+esi*4]
    add    edx, [ebx+esi*4]
    mov    [eax+esi*4], edx
    inc    esi
    cmp    esi, edi
    jb     short loc_C1
```

```
    mov      ecx, [esp+10h+var_10]
    mov      edx, [esp+10h+sz]

loc_D6: ; CODE XREF: f(int,int *,int *,int *)+B2
    mov      esi, [esp+10h+ar2]
    lea      esi, [esi+edi*4] ; ar2+i*4 ?
    test     esi, 0Fh
    jz      short loc_109   ; !
    mov      ebx, [esp+10h+ar1]
    mov      esi, [esp+10h+ar2]

loc_ED: ; CODE XREF: f(int,int *,int *,int *)+105
    movdqu  xmm1, xmmword ptr [ebx+edi*4] ; ar1+i*4
    movdqu  xmm0, xmmword ptr [esi+edi*4] ; ar2+i*4  XMM0
    padddd  xmm1, xmm0
    movdqa  xmmword ptr [eax+edi*4], xmm1 ; ar3+i*4
    add     edi, 4
    cmp     edi, ecx
    jb      short loc_ED
    jmp     short loc_127

loc_109: ; CODE XREF: f(int,int *,int *,int *)+E3
    mov      ebx, [esp+10h+ar1]
    mov      esi, [esp+10h+ar2]

loc_111: ; CODE XREF: f(int,int *,int *,int *)+125
    movdqu  xmm0, xmmword ptr [ebx+edi*4]
    padddd  xmm0, xmmword ptr [esi+edi*4]
    movdqa  xmmword ptr [eax+edi*4], xmm0
    add     edi, 4
    cmp     edi, ecx
    jb      short loc_111

loc_127: ; CODE XREF: f(int,int *,int *,int *)+107
           ; f(int,int *,int *,int *)+164
    cmp     ecx, edx
    jnb     short loc_15B
    mov      esi, [esp+10h+ar1]
    mov      edi, [esp+10h+ar2]

loc_133: ; CODE XREF: f(int,int *,int *,int *)+13F
    mov      ebx, [esi+ecx*4]
    add     ebx, [edi+ecx*4]
    mov      [eax+ecx*4], ebx
    inc     ecx
    cmp     ecx, edx
    jb      short loc_133
```

```

        jmp      short loc_15B

loc_143: ; CODE XREF: f(int,int *,int *,int *)+17
          ; f(int,int *,int *,int *)+3A ...
        mov      esi, [esp+10h+ar1]
        mov      edi, [esp+10h+ar2]
        xor      ecx, ecx

loc_14D: ; CODE XREF: f(int,int *,int *,int *)+159
        mov      ebx, [esi+ecx*4]
        add      ebx, [edi+ecx*4]
        mov      [eax+ecx*4], ebx
        inc      ecx
        cmp      ecx, edx
        jb       short loc_14D

loc_15B: ; CODE XREF: f(int,int *,int *,int *)+A
          ; f(int,int *,int *,int *)+129 ...
        xor      eax, eax
        pop      ecx
        pop      ebx
        pop      esi
        pop      edi
        retn

loc_162: ; CODE XREF: f(int,int *,int *,int *)+8C
          ; f(int,int *,int *,int *)+9F
        xor      ecx, ecx
        jmp      short loc_127
?f@@YAHHPAH00@Z endp

```

- **MOVDQU** (*Move Unaligned Double Quadword*) .
- **PADDD** (*Add Packed Integers*)
- **MOVDQA** (*Move Aligned Double Quadword*)

<pre> movdqu xmm0, xmmword ptr [ebx+edi*4] ; ar1+i*4 paddd xmm0, xmmword ptr [esi+edi*4] ; ar2+i*4 movdqa xmmword ptr [eax+edi*4], xmm0 ; ar3+i*4 </pre>
--

<pre> movdqu xmm1, xmmword ptr [ebx+edi*4] ; ar1+i*4 movdqu xmm0, xmmword ptr [esi+edi*4] ; ar2+i*4 XMMO paddd xmm1, xmm0 movdqa xmmword ptr [eax+edi*4], xmm1 ; ar3+i*4 </pre>

GCC

(GCC 4.4.1):

```
; f(int, int *, int *, int *)
    public _Z1fiPiS_S_
_Z1fiPiS_S_ proc near

var_18      = dword ptr -18h
var_14      = dword ptr -14h
var_10      = dword ptr -10h
arg_0       = dword ptr  8
arg_4       = dword ptr  0Ch
arg_8       = dword ptr  10h
arg_C       = dword ptr  14h

push        ebp
mov         ebp, esp
push        edi
push        esi
push        ebx
sub         esp, 0Ch
mov         ecx, [ebp+arg_0]
mov         esi, [ebp+arg_4]
mov         edi, [ebp+arg_8]
mov         ebx, [ebp+arg_C]
test        ecx, ecx
jle         short loc_80484D8
cmp         ecx, 6
lea         eax, [ebx+10h]
ja          short loc_80484E8

loc_80484C1: ; CODE XREF: f(int,int *,int *,int *)+4B
              ; f(int,int *,int *,int *)+61 ...
xor         eax, eax
nop
lea         esi, [esi+0]

loc_80484C8: ; CODE XREF: f(int,int *,int *,int *)+36
mov         edx, [edi+eax*4]
add         edx, [esi+eax*4]
mov         [ebx+eax*4], edx
add         eax, 1
cmp         eax, ecx
jnz         short loc_80484C8

loc_80484D8: ; CODE XREF: f(int,int *,int *,int *)+17
              ; f(int,int *,int *,int *)+A5
```

```
    add    esp, 0Ch
    xor    eax, eax
    pop    ebx
    pop    esi
    pop    edi
    pop    ebp
    retn

    align 8

loc_80484E8: ; CODE XREF: f(int,int *,int *,int *)+1F
    test   bl, 0Fh
    jnz    short loc_80484C1
    lea    edx, [esi+10h]
    cmp    ebx, edx
    jbe    loc_8048578

loc_80484F8: ; CODE XREF: f(int,int *,int *,int *)+E0
    lea    edx, [edi+10h]
    cmp    ebx, edx
    ja     short loc_8048503
    cmp    edi, eax
    jbe    short loc_80484C1

loc_8048503: ; CODE XREF: f(int,int *,int *,int *)+5D
    mov    eax, ecx
    shr    eax, 2
    mov    [ebp+var_14], eax
    shl    eax, 2
    test   eax, eax
    mov    [ebp+var_10], eax
    jz    short loc_8048547
    mov    [ebp+var_18], ecx
    mov    ecx, [ebp+var_14]
    xor    eax, eax
    xor    edx, edx
    nop

loc_8048520: ; CODE XREF: f(int,int *,int *,int *)+9B
    movdqu xmm1, xmmword ptr [edi+eax]
    movdqu xmm0, xmmword ptr [esi+eax]
    add    edx, 1
    paddd xmm0, xmm1
    movdqa xmmword ptr [ebx+eax], xmm0
    add    eax, 10h
    cmp    edx, ecx
    jb     short loc_8048520
```

```

        mov      ecx, [ebp+var_18]
        mov      eax, [ebp+var_10]
        cmp      ecx, eax
        jz       short loc_80484D8

loc_8048547: ; CODE XREF: f(int,int *,int *,int *)+73
        lea      edx, ds:0[eax*4]
        add      esi, edx
        add      edi, edx
        add      ebx, edx
        lea      esi, [esi+0]

loc_8048558: ; CODE XREF: f(int,int *,int *,int *)+CC
        mov      edx, [edi]
        add      eax, 1
        add      edi, 4
        add      edx, [esi]
        add      esi, 4
        mov      [ebx], edx
        add      ebx, 4
        cmp      ecx, eax
        jg      short loc_8048558
        add      esp, 0Ch
        xor      eax, eax
        pop     ebx
        pop     esi
        pop     edi
        pop     ebp
        retn

loc_8048578: ; CODE XREF: f(int,int *,int *,int *)+52
        cmp      eax, esi
        jnb      loc_80484C1
        jmp      loc_80484F8
_Z1fiPiS_S_ endp

```

25.1.2

([14.2 on page 233](#)):

```
#include <stdio.h>

void my_memcpy (unsigned char* dst, unsigned char* src, size_t ↴
    ↴ cnt)
{
    size_t i;
    for (i=0; i<cnt; i++)

```

```
        dst[i]=src[i];  
};
```

Listing 25.1: Con optimización GCC 4.9.1 x64

```
my_memcpy:  
; RDI =  
; RSI =  
; RDX =  
    test    rdx, rdx  
    je      .L41  
    lea     rax, [rdi+16]  
    cmp     rsi, rax  
    lea     rax, [rsi+16]  
    setae   cl  
    cmp     rdi, rax  
    setae   al  
    or      cl, al  
    je      .L13  
    cmp     rdx, 22  
    jbe     .L13  
    mov     rcx, rsi  
    push    rbp  
    push    rbx  
    neg     rcx  
    and    ecx, 15  
    cmp     rcx, rdx  
    cmova  rcx, rdx  
    xor     eax, eax  
    test    rcx, rcx  
    je      .L4  
    movzx   eax, BYTE PTR [rsi]  
    cmp     rcx, 1  
    mov     BYTE PTR [rdi], al  
    je      .L15  
    movzx   eax, BYTE PTR [rsi+1]  
    cmp     rcx, 2  
    mov     BYTE PTR [rdi+1], al  
    je      .L16  
    movzx   eax, BYTE PTR [rsi+2]  
    cmp     rcx, 3  
    mov     BYTE PTR [rdi+2], al  
    je      .L17  
    movzx   eax, BYTE PTR [rsi+3]  
    cmp     rcx, 4  
    mov     BYTE PTR [rdi+3], al  
    je      .L18
```

```
    movzx  eax, BYTE PTR [rsi+4]
    cmp    rcx, 5
    mov    BYTE PTR [rdi+4], al
    je     .L19
    movzx  eax, BYTE PTR [rsi+5]
    cmp    rcx, 6
    mov    BYTE PTR [rdi+5], al
    je     .L20
    movzx  eax, BYTE PTR [rsi+6]
    cmp    rcx, 7
    mov    BYTE PTR [rdi+6], al
    je     .L21
    movzx  eax, BYTE PTR [rsi+7]
    cmp    rcx, 8
    mov    BYTE PTR [rdi+7], al
    je     .L22
    movzx  eax, BYTE PTR [rsi+8]
    cmp    rcx, 9
    mov    BYTE PTR [rdi+8], al
    je     .L23
    movzx  eax, BYTE PTR [rsi+9]
    cmp    rcx, 10
    mov   BYTE PTR [rdi+9], al
    je     .L24
    movzx  eax, BYTE PTR [rsi+10]
    cmp   rcx, 11
    mov   BYTE PTR [rdi+10], al
    je     .L25
    movzx  eax, BYTE PTR [rsi+11]
    cmp   rcx, 12
    mov   BYTE PTR [rdi+11], al
    je     .L26
    movzx  eax, BYTE PTR [rsi+12]
    cmp   rcx, 13
    mov   BYTE PTR [rdi+12], al
    je     .L27
    movzx  eax, BYTE PTR [rsi+13]
    cmp   rcx, 15
    mov   BYTE PTR [rdi+13], al
    jne    .L28
    movzx  eax, BYTE PTR [rsi+14]
    mov   BYTE PTR [rdi+14], al
    mov    eax, 15
.L4:
    mov    r10, rdx
    lea    r9, [rdx-1]
    sub    r10, rcx
```

```

    lea    r8, [r10-16]
    sub    r9, rcx
    shr    r8, 4
    add    r8, 1
    mov    r11, r8
    sal    r11, 4
    cmp    r9, 14
    jbe    .L6
    lea    rbp, [rsi+rcx]
    xor    r9d, r9d
    add    rcx, rdi
    xor    ebx, ebx

.L7:
    movdqa xmm0, XMMWORD PTR [rbp+0+r9]
    add    rbx, 1
    movups XMMWORD PTR [rcx+r9], xmm0
    add    r9, 16
    cmp    rbx, r8
    jb     .L7
    add    rax, r11
    cmp    r10, r11
    je     .L1

.L6:
    movzx  ecx, BYTE PTR [rsi+rax]
    mov    BYTE PTR [rdi+rax], cl
    lea    rcx, [rax+1]
    cmp    rdx, rcx
    jbe    .L1
    movzx  ecx, BYTE PTR [rsi+1+rax]
    mov    BYTE PTR [rdi+1+rax], cl
    lea    rcx, [rax+2]
    cmp    rdx, rcx
    jbe    .L1
    movzx  ecx, BYTE PTR [rsi+2+rax]
    mov    BYTE PTR [rdi+2+rax], cl
    lea    rcx, [rax+3]
    cmp    rdx, rcx
    jbe    .L1
    movzx  ecx, BYTE PTR [rsi+3+rax]
    mov    BYTE PTR [rdi+3+rax], cl
    lea    rcx, [rax+4]
    cmp    rdx, rcx
    jbe    .L1
    movzx  ecx, BYTE PTR [rsi+4+rax]
    mov    BYTE PTR [rdi+4+rax], cl
    lea    rcx, [rax+5]
    cmp    rdx, rcx

```

```
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+5+rax]
mov    BYTE PTR [rdi+5+rax], cl
lea    rcx,  [rax+6]
cmp    rdx,  rcx
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+6+rax]
mov    BYTE PTR [rdi+6+rax], cl
lea    rcx,  [rax+7]
cmp    rdx,  rcx
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+7+rax]
mov    BYTE PTR [rdi+7+rax], cl
lea    rcx,  [rax+8]
cmp    rdx,  rcx
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+8+rax]
mov    BYTE PTR [rdi+8+rax], cl
lea    rcx,  [rax+9]
cmp    rdx,  rcx
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+9+rax]
mov    BYTE PTR [rdi+9+rax], cl
lea    rcx,  [rax+10]
cmp    rdx,  rcx
jbe    .L1
movzx  ecx,  BYTE PTR [rsi+10+rax]
mov   BYTE PTR [rdi+10+rax], cl
lea   rcx,  [rax+11]
cmp   rdx,  rcx
jbe   .L1
movzx  ecx,  BYTE PTR [rsi+11+rax]
mov   BYTE PTR [rdi+11+rax], cl
lea   rcx,  [rax+12]
cmp   rdx,  rcx
jbe   .L1
movzx  ecx,  BYTE PTR [rsi+12+rax]
mov   BYTE PTR [rdi+12+rax], cl
lea   rcx,  [rax+13]
cmp   rdx,  rcx
jbe   .L1
movzx  ecx,  BYTE PTR [rsi+13+rax]
mov   BYTE PTR [rdi+13+rax], cl
lea   rcx,  [rax+14]
cmp   rdx,  rcx
jbe   .L1
movzx  edx,  BYTE PTR [rsi+14+rax]
```

```
    mov     BYTE PTR [rdi+14+rax], dl
.L1:
    pop     rbx
    pop     rbp
.L41:
    rep ret
.L13:
    xor     eax, eax
.L3:
    movzx  ecx, BYTE PTR [rsi+rax]
    mov     BYTE PTR [rdi+rax], cl
    add     rax, 1
    cmp     rax, rdx
    jne     .L3
    rep ret
.L28:
    mov     eax, 14
    jmp     .L4
.L15:
    mov     eax, 1
    jmp     .L4
.L16:
    mov     eax, 2
    jmp     .L4
.L17:
    mov     eax, 3
    jmp     .L4
.L18:
    mov     eax, 4
    jmp     .L4
.L19:
    mov     eax, 5
    jmp     .L4
.L20:
    mov     eax, 6
    jmp     .L4
.L21:
    mov     eax, 7
    jmp     .L4
.L22:
    mov     eax, 8
    jmp     .L4
.L23:
    mov     eax, 9
    jmp     .L4
.L24:
    mov     eax, 10
```

```

        jmp     .L4
.L25:
    mov     eax, 11
    jmp     .L4
.L26:
    mov     eax, 12
    jmp     .L4
.L27:
    mov     eax, 13
    jmp     .L4

```

25.2

[2.](#)

```

size_t strlen_sse2(const char *str)
{
    register size_t len = 0;
    const char *s=str;
    bool str_is_aligned=((unsigned int)str)&0xFFFFFFFF0) == ((unsigned int)str);

    if (str_is_aligned==false)
        return strlen (str);

    __m128i xmm0 = _mm_setzero_si128();
    __m128i xmm1;
    int mask = 0;

    for (;;)
    {
        xmm1 = _mm_load_si128((__m128i *)s);
        xmm1 = _mm_cmpeq_epi8(xmm1, xmm0);
        if ((mask = _mm_movemask_epi8(xmm1)) != 0)
        {
            unsigned long pos;
            _BitScanForward(&pos, mask);
            len += (size_t)pos;

            break;
        }
        s += sizeof(__m128i);
        len += sizeof(__m128i);
    };
}

```

```
    return len;
}
```

Listing 25.2: Con optimización MSVC 2010

```
_pos$75552 = -4          ; size = 4
_str$ = 8                ; size = 4
?strlen_sse2@@YAIPBD@Z PROC ; strlen_sse2

    push    ebp
    mov     ebp, esp
    and     esp, -16           ; ffffffff0H
    mov     eax, DWORD PTR _str$[ebp]
    sub     esp, 12            ; 0000000cH
    push    esi
    mov     esi, eax
    and     esi, -16           ; ffffffff0H
    xor     edx, edx
    mov     ecx, eax
    cmp     esi, eax
    je      SHORT $LN4@strlen_sse
    lea     edx, DWORD PTR [eax+1]
    npad   3 ;
$LL11@strlen_sse:
    mov     cl, BYTE PTR [eax]
    inc     eax
    test   cl, cl
    jne   SHORT $LL11@strlen_sse
    sub     eax, edx
    pop     esi
    mov     esp, ebp
    pop     ebp
    ret     0
$LN4@strlen_sse:
    movdqa  xmm1, XMMWORD PTR [eax]
    pxor    xmm0, xmm0
    pcmpeqb xmm1, xmm0
    pmovmskb eax, xmm1
    test    eax, eax
    jne   SHORT $LN9@strlen_sse
$LL3@strlen_sse:
    movdqa  xmm1, XMMWORD PTR [ecx+16]
    add    ecx, 16             ; 00000010H
    pcmpeqb xmm1, xmm0
    add    edx, 16             ; 00000010H
    pmovmskb eax, xmm1
```

```

    test    eax, eax
    je      SHORT $LL3@strlen_sse
$LN9@strlen_sse:
    bsf    eax, eax
    mov    ecx, eax
    mov    DWORD PTR _pos$75552[esp+16], eax
    lea    eax, DWORD PTR [ecx+edx]
    pop    esi
    mov    esp, ebp
    pop    ebp
    ret    0
?strlen_sse2@@YAIPBD@Z ENDP ; strlen_sse2

```

0x008c1ff8	'h'
0x008c1ff9	'e'
0x008c1ffa	'l'
0x008c1ffb	'l'
0x008c1ffc	'o'
0x008c1ffd	'\x00'
0x008c1ffe	
0x008c1fff	

_mm_setzero_si128().

_mm_load_si128()

_mm_cmpeq_epi8()

XMM1: 112233445566778800000000000000000000

XMM0: 11ab34440078778811111111111111111111

XMM1: ff0000ff0000ffff00000000000000000000

pmovmskb eax, xmm1

:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
«h»	«e»	«l»	«l»	«o»	0		Basura		0	Basura					

XMM1: 0000ff00000000000000ff0000000000

0010000000100000b.

BSF (Bit Scan Forward).

EAX=0010000000100000b

_BitScanForward.

<http://go.yurichev.com/17331>

Capítulo 26

26.1 x86-64

- RAX, RBX, RCX, RDX, RBP, RSP, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15.

Spanish text placeholder	
RAX ^{x64}	
	EAX
	AX
AH	AL

Spanish text placeholder	
R8	
	R8D
	R8W
	R8L

XMM0-XMM15.

- System V AMD64 ABI (Linux, *BSD, Mac OS X)[[Mit13](#)] fastcall, RDI, RSI, RDX, RCX, R8, R9 ..

([64 on page 869](#)).

-
- .

```
/*
 * Generated S-box files.
 *
 * This software may be modified, redistributed, and used for ↴
 *   ↴ any purpose,
 * so long as its origin is acknowledged.
```

```
*  
* Produced by Matthew Kwan - March 1998  
*/  
  
#ifdef _WIN64  
#define DES_type unsigned __int64  
#else  
#define DES_type unsigned int  
#endif  
  
void  
s1 (  
    DES_type    a1,  
    DES_type    a2,  
    DES_type    a3,  
    DES_type    a4,  
    DES_type    a5,  
    DES_type    a6,  
    DES_type    *out1,  
    DES_type    *out2,  
    DES_type    *out3,  
    DES_type    *out4  
) {  
    DES_type    x1, x2, x3, x4, x5, x6, x7, x8;  
    DES_type    x9, x10, x11, x12, x13, x14, x15, x16;  
    DES_type    x17, x18, x19, x20, x21, x22, x23, x24;  
    DES_type    x25, x26, x27, x28, x29, x30, x31, x32;  
    DES_type    x33, x34, x35, x36, x37, x38, x39, x40;  
    DES_type    x41, x42, x43, x44, x45, x46, x47, x48;  
    DES_type    x49, x50, x51, x52, x53, x54, x55, x56;  
  
    x1 = a3 & ~a5;  
    x2 = x1 ^ a4;  
    x3 = a3 & ~a4;  
    x4 = x3 | a5;  
    x5 = a6 & x4;  
    x6 = x2 ^ x5;  
    x7 = a4 & ~a5;  
    x8 = a3 ^ a4;  
    x9 = a6 & ~x8;  
    x10 = x7 ^ x9;  
    x11 = a2 | x10;  
    x12 = x6 ^ x11;  
    x13 = a5 ^ x5;  
    x14 = x13 & x8;  
    x15 = a5 & ~a4;  
    x16 = x3 ^ x14;
```

```
x17 = a6 | x16;
x18 = x15 ^ x17;
x19 = a2 | x18;
x20 = x14 ^ x19;
x21 = a1 & x20;
x22 = x12 ^ ~x21;
*out2 ^= x22;
x23 = x1 | x5;
x24 = x23 ^ x8;
x25 = x18 & ~x2;
x26 = a2 & ~x25;
x27 = x24 ^ x26;
x28 = x6 | x7;
x29 = x28 ^ x25;
x30 = x9 ^ x24;
x31 = x18 & ~x30;
x32 = a2 & x31;
x33 = x29 ^ x32;
x34 = a1 & x33;
x35 = x27 ^ x34;
*out4 ^= x35;
x36 = a3 & x28;
x37 = x18 & ~x36;
x38 = a2 | x3;
x39 = x37 ^ x38;
x40 = a3 | x31;
x41 = x24 & ~x37;
x42 = x41 | x3;
x43 = x42 & ~a2;
x44 = x40 ^ x43;
x45 = a1 & ~x44;
x46 = x39 ^ ~x45;
*out1 ^= x46;
x47 = x33 & ~x9;
x48 = x47 ^ x39;
x49 = x4 & x36;
x50 = x49 & ~x5;
x51 = x42 | x18;
x52 = x51 ^ a5;
x53 = a2 & ~x52;
x54 = x50 ^ x53;
x55 = a1 | x54;
x56 = x48 ^ ~x55;
*out3 ^= x56;
}
```

Listing 26.1: Con optimización MSVC 2008

```

PUBLIC    _s1
; Function compile flags: /Ogtpy
_TEXT     SEGMENT
_x6$ = -20           ; size = 4
_x3$ = -16           ; size = 4
_x1$ = -12           ; size = 4
_x8$ = -8            ; size = 4
_x4$ = -4            ; size = 4
_a1$ = 8             ; size = 4
_a2$ = 12            ; size = 4
_a3$ = 16            ; size = 4
_x33$ = 20           ; size = 4
_x7$ = 20             ; size = 4
_a4$ = 20             ; size = 4
_a5$ = 24             ; size = 4
tv326 = 28            ; size = 4
_x36$ = 28            ; size = 4
_x28$ = 28            ; size = 4
_a6$ = 28             ; size = 4
_out1$ = 32            ; size = 4
_x24$ = 36             ; size = 4
_out2$ = 36             ; size = 4
_out3$ = 40             ; size = 4
_out4$ = 44             ; size = 4
_s1    PROC
        sub    esp, 20          ; 00000014H
        mov    edx, DWORD PTR _a5$[esp+16]
        push   ebx
        mov    ebx, DWORD PTR _a4$[esp+20]
        push   ebp
        push   esi
        mov    esi, DWORD PTR _a3$[esp+28]
        push   edi
        mov    edi, ebx
        not    edi
        mov    ebp, edi
        and    edi, DWORD PTR _a5$[esp+32]
        mov    ecx, edx
        not    ecx
        and    ebp, esi
        mov    eax, ecx
        and    eax, esi
        and    ecx, ebx
        mov    DWORD PTR _x1$[esp+36], eax
        xor    eax, ebx
        mov    esi, ebp

```

```
or    esi, edx
mov   DWORD PTR _x4$[esp+36], esi
and   esi, DWORD PTR _a6$[esp+32]
mov   DWORD PTR _x7$[esp+32], ecx
mov   edx, esi
xor   edx, eax
mov   DWORD PTR _x6$[esp+36], edx
mov   edx, DWORD PTR _a3$[esp+32]
xor   edx, ebx
mov   ebx, esi
xor   ebx, DWORD PTR _a5$[esp+32]
mov   DWORD PTR _x8$[esp+36], edx
and   ebx, edx
mov   ecx, edx
mov   edx, ebx
xor   edx, ebp
or    edx, DWORD PTR _a6$[esp+32]
not   ecx
and   ecx, DWORD PTR _a6$[esp+32]
xor   edx, edi
mov   edi, edx
or    edi, DWORD PTR _a2$[esp+32]
mov   DWORD PTR _x3$[esp+36], ebp
mov   ebp, DWORD PTR _a2$[esp+32]
xor   edi, ebx
and   edi, DWORD PTR _a1$[esp+32]
mov   ebx, ecx
xor   ebx, DWORD PTR _x7$[esp+32]
not   edi
or    ebx, ebp
xor   edi, ebx
mov   ebx, edi
mov   edi, DWORD PTR _out2$[esp+32]
xor   ebx, DWORD PTR [edi]
not   eax
xor   ebx, DWORD PTR _x6$[esp+36]
and   eax, edx
mov   DWORD PTR [edi], ebx
mov   ebx, DWORD PTR _x7$[esp+32]
or    ebx, DWORD PTR _x6$[esp+36]
mov   edi, esi
or    edi, DWORD PTR _x1$[esp+36]
mov   DWORD PTR _x28$[esp+32], ebx
xor   edi, DWORD PTR _x8$[esp+36]
mov   DWORD PTR _x24$[esp+32], edi
xor   edi, ecx
not   edi
```

```
and    edi, edx
mov    ebx, edi
and    ebx, ebp
xor    ebx, DWORD PTR _x28$[esp+32]
xor    ebx, eax
not    eax
mov    DWORD PTR _x33$[esp+32], ebx
and    ebx, DWORD PTR _a1$[esp+32]
and    eax, ebp
xor    eax, ebx
mov    ebx, DWORD PTR _out4$[esp+32]
xor    eax, DWORD PTR [ebx]
xor    eax, DWORD PTR _x24$[esp+32]
mov    DWORD PTR [ebx], eax
mov    eax, DWORD PTR _x28$[esp+32]
and    eax, DWORD PTR _a3$[esp+32]
mov    ebx, DWORD PTR _x3$[esp+36]
or     edi, DWORD PTR _a3$[esp+32]
mov    DWORD PTR _x36$[esp+32], eax
not    eax
and    eax, edx
or     ebx, ebp
xor    ebx, eax
not    eax
and    eax, DWORD PTR _x24$[esp+32]
not    ebp
or     eax, DWORD PTR _x3$[esp+36]
not    esi
and    ebp, eax
or     eax, edx
xor    eax, DWORD PTR _a5$[esp+32]
mov    edx, DWORD PTR _x36$[esp+32]
xor    edx, DWORD PTR _x4$[esp+36]
xor    ebp, edi
mov    edi, DWORD PTR _out1$[esp+32]
not    eax
and    eax, DWORD PTR _a2$[esp+32]
not    ebp
and    ebp, DWORD PTR _a1$[esp+32]
and    edx, esi
xor    eax, edx
or     eax, DWORD PTR _a1$[esp+32]
not    ebp
xor    ebp, DWORD PTR [edi]
not    ecx
and    ecx, DWORD PTR _x33$[esp+32]
xor    ebp, ebx
```

```

not    eax
mov    DWORD PTR [edi], ebp
xor    eax, ecx
mov    ecx, DWORD PTR _out3$[esp+32]
xor    eax, DWORD PTR [ecx]
pop    edi
pop    esi
xor    eax, ebx
pop    ebp
mov    DWORD PTR [ecx], eax
pop    ebx
add    esp, 20           ; 00000014H
ret    0
_s1    ENDP

```

Listing 26.2: Con optimización MSVC 2008

```

a1$ = 56
a2$ = 64
a3$ = 72
a4$ = 80
x36$1$ = 88
a5$ = 88
a6$ = 96
out1$ = 104
out2$ = 112
out3$ = 120
out4$ = 128
s1    PROC
$LN3:
    mov    QWORD PTR [rsp+24], rbx
    mov    QWORD PTR [rsp+32], rbp
    mov    QWORD PTR [rsp+16], rdx
    mov    QWORD PTR [rsp+8], rcx
    push   rsi
    push   rdi
    push   r12
    push   r13
    push   r14
    push   r15
    mov    r15, QWORD PTR a5$[rsp]
    mov    rcx, QWORD PTR a6$[rsp]
    mov    rbp, r8
    mov    r10, r9
    mov    rax, r15
    mov    rdx, rbp
    not    rax
    xor    rdx, r9

```

```
not    r10
mov    r11, rax
and    rax, r9
mov    rsi, r10
mov    QWORD PTR x36$1$[rsp], rax
and    r11, r8
and    rsi, r8
and    r10, r15
mov    r13, rdx
mov    rbx, r11
xor    rbx, r9
mov    r9, QWORD PTR a2$[rsp]
mov    r12, rsi
or     r12, r15
not    r13
and    r13, rcx
mov    r14, r12
and    r14, rcx
mov    rax, r14
mov    r8, r14
xor    r8, rbx
xor    rax, r15
not    rbx
and    rax, rdx
mov    rdi, rax
xor    rdi, rsi
or     rdi, rcx
xor    rdi, r10
and    rbx, rdi
mov    rcx, rdi
or     rcx, r9
xor    rcx, rax
mov    rax, r13
xor    rax, QWORD PTR x36$1$[rsp]
and    rcx, QWORD PTR a1$[rsp]
or     rax, r9
not    rcx
xor    rcx, rax
mov    rax, QWORD PTR out2$[rsp]
xor    rcx, QWORD PTR [rax]
xor    rcx, r8
mov    QWORD PTR [rax], rcx
mov    rax, QWORD PTR x36$1$[rsp]
mov    rcx, r14
or     rax, r8
or     rcx, r11
mov    r11, r9
```

```
xor    rcx, rdx
mov    QWORD PTR x36$1$[rsp], rax
mov    r8, rsi
mov    rdx, rcx
xor    rdx, r13
not    rdx
and    rdx, rdi
mov    r10, rdx
and    r10, r9
xor    r10, rax
xor    r10, rbx
not    rbx
and    rbx, r9
mov    rax, r10
and    rax, QWORD PTR a1$[rsp]
xor    rbx, rax
mov    rax, QWORD PTR out4$[rsp]
xor    rbx, QWORD PTR [rax]
xor    rbx, rcx
mov    QWORD PTR [rax], rbx
mov    rbx, QWORD PTR x36$1$[rsp]
and    rbx, rbp
mov    r9, rbx
not    r9
and    r9, rdi
or     r8, r11
mov    rax, QWORD PTR out1$[rsp]
xor    r8, r9
not    r9
and    r9, rcx
or     rdx, rbp
mov    rbp, QWORD PTR [rsp+80]
or     r9, rsi
xor    rbx, r12
mov    rcx, r11
not    rcx
not    r14
not    r13
and    rcx, r9
or     r9, rdi
and    rbx, r14
xor    r9, r15
xor    rcx, rdx
mov    rdx, QWORD PTR a1$[rsp]
not    r9
not    rcx
and    r13, r10
```

```
and    r9, r11
and    rcx, rdx
xor    r9, rbx
mov    rbx, QWORD PTR [rsp+72]
not    rcx
xor    rcx, QWORD PTR [rax]
or     r9, rdx
not    r9
xor    rcx, r8
mov    QWORD PTR [rax], rcx
mov    rax, QWORD PTR out3$[rsp]
xor    r9, r13
xor    r9, QWORD PTR [rax]
xor    r9, r8
mov    QWORD PTR [rax], r9
pop    r15
pop    r14
pop    r13
pop    r12
pop    rdi
pop    rsi
ret    0
s1    ENDP
```

26.2 ARM

ARMv8.

26.3

Capítulo 27

SIMD (SSE2).

(IEEE 754).

: 17 on page 270.

27.1

```
#include <stdio.h>

double f (double a, double b)
{
    return a/3.14 + b*4.1;
}

int main()
{
    printf ("%f\n", f(1.2, 3.4));
}
```

27.1.1 x64

Listing 27.1: Con optimización MSVC 2012 x64

```
__real@4010666666666666 DQ 0401066666666666r ; 4.1
__real@40091eb851eb851f DQ 040091eb851eb851fr ; 3.14
a$ = 8
```

```
b$ = 16
f      PROC
        divsd  xmm0, QWORD PTR __real@40091eb851eb851f
        mulsd  xmm1, QWORD PTR __real@4010666666666666
        addsd  xmm0, xmm1
        ret    0
f      ENDP
```

[1](#).

a XMM0, b XMM1. .

DIVSD «Divide Scalar Double-Precision Floating-Point Values», .

MULSD y ADDSD .

.

:

Listing 27.2: MSVC 2012 x64

```
__real@4010666666666666 DQ 0401066666666666r ; 4.1
__real@40091eb851eb851f DQ 040091eb851eb851fr ; 3.14

a$ = 8
b$ = 16
f      PROC
        movsd  QWORD PTR [rsp+16], xmm1
        movsd  QWORD PTR [rsp+8], xmm0
        movsd  xmm0, QWORD PTR a$[rsp]
        divsd  xmm0, QWORD PTR __real@40091eb851eb851f
        movsd  xmm1, QWORD PTR b$[rsp]
        mulsd  xmm1, QWORD PTR __real@4010666666666666
        addsd  xmm0, xmm1
        ret    0
f      ENDP
```

. «shadow space» ([8.2.1 on page 118](#)), .

27.1.2 x86

Listing 27.3: Sin optimización MSVC 2012 x86

```
tv70 = -8          ; size = 8
```

¹[MSDN: Parameter Passing](#)

```

_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_f      PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 8
    movsd   xmm0, QWORD PTR _a$[ebp]
    divsd   xmm0, QWORD PTR __real@40091eb851eb851f
    movsd   xmm1, QWORD PTR _b$[ebp]
    mulsd   xmm1, QWORD PTR __real@4010666666666666
    addsd   xmm0, xmm1
    movsd   QWORD PTR tv70[ebp], xmm0
    fld     QWORD PTR tv70[ebp]
    mov     esp, ebp
    pop     ebp
    ret     0
_f      ENDP

```

Listing 27.4: Con optimización MSVC 2012 x86

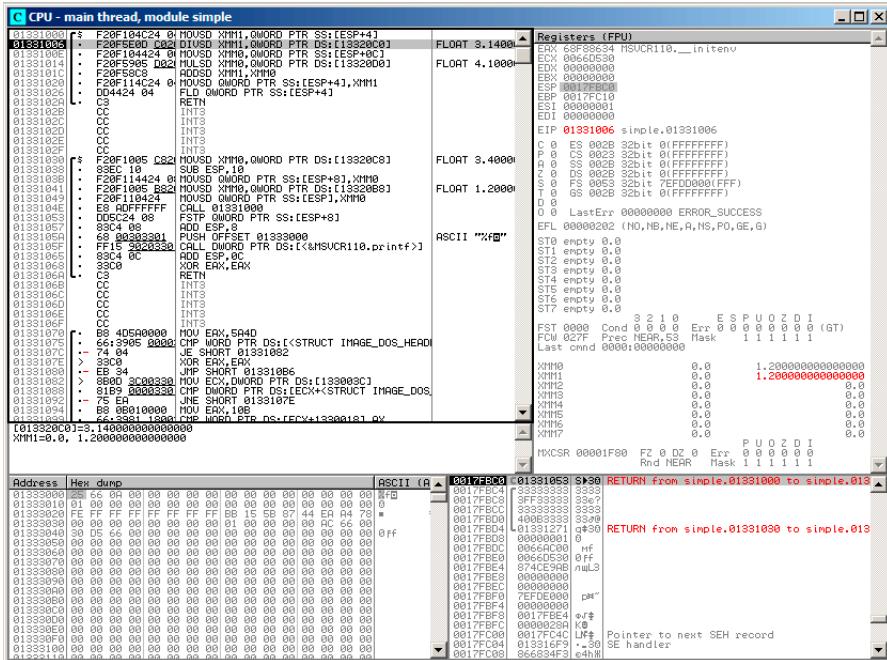
```

tv67 = 8          ; size = 8
_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_f      PROC
    movsd   xmm1, QWORD PTR _a$[esp-4]
    divsd   xmm1, QWORD PTR __real@40091eb851eb851f
    movsd   xmm0, QWORD PTR _b$[esp-4]
    mulsd   xmm0, QWORD PTR __real@4010666666666666
    addsd   xmm1, xmm0
    movsd   QWORD PTR tv67[esp-4], xmm1
    fld     QWORD PTR tv67[esp-4]
    ret     0
_f      ENDP

```

1) 2)

OllyDbg:



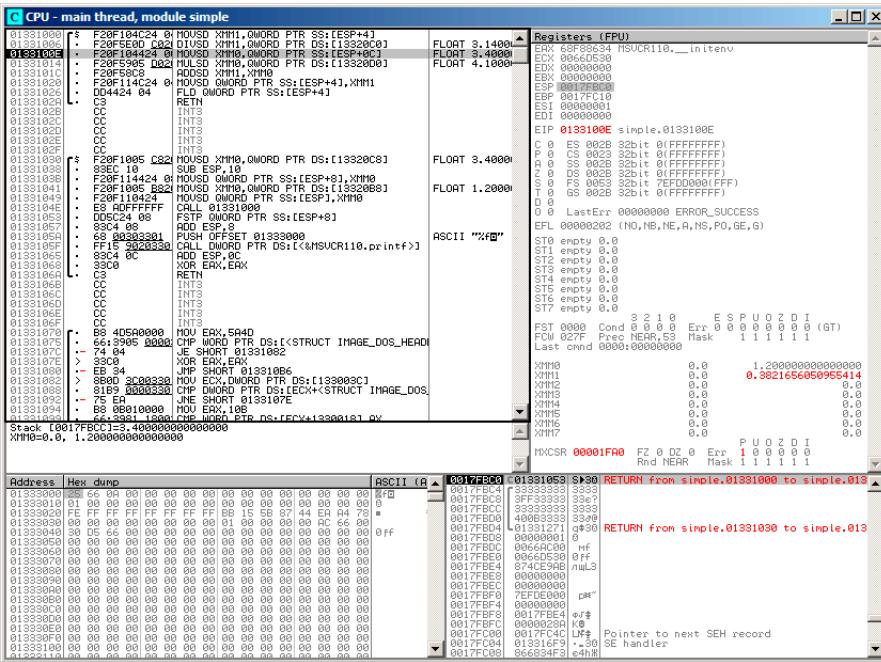


Figura 27.2: OllyDbg: DIVSD quotient XMM1

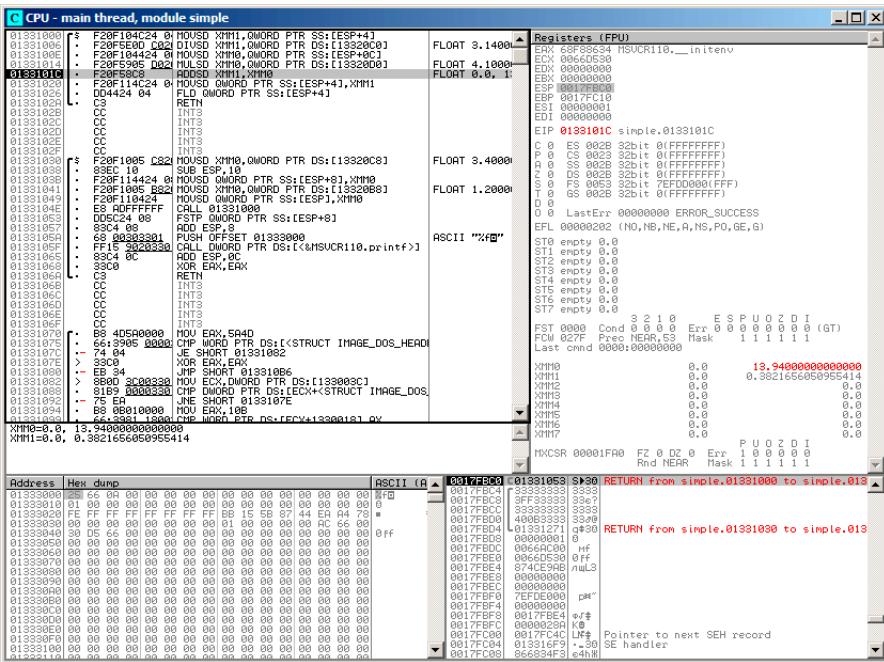


Figura 27.3: OllyDbg: MULSD product XMM0

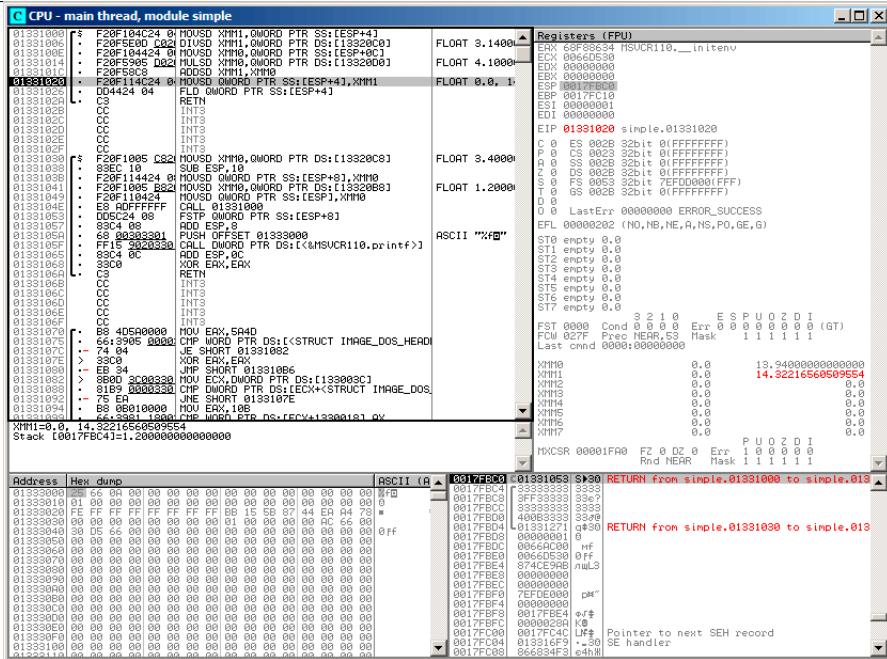


Figura 27.4: OllyDbg: ADDSD XMMO XM1

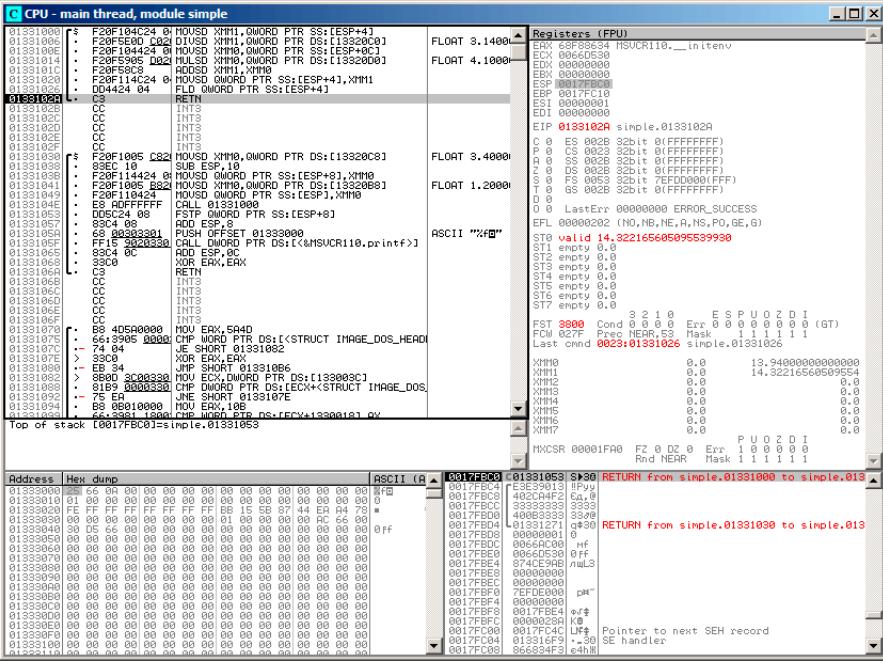


Figura 27.5: OllyDbg: FLD ST(0)

27.2

```
#include <math.h>
#include <stdio.h>

int main ()
{
    printf ("32.01 ^ 1.54 = %lf\n", pow (32.01,1.54));

    return 0;
}
```

XMM0-XMM3.

Listing 27.5: Con optimización MSVC 2012 x64

```
$SG1354 DB      '32.01 ^ 1.54 = %lf', 0AH, 00H

__real@40400147ae147ae1 DQ 040400147ae147ae1r ; 32.01
__real@3ff8a3d70a3d70a4 DQ 03ff8a3d70a3d70a4r ; 1.54

main    PROC
        sub     rsp, 40
        ↳ 00000028H
        movsdx xmm1, QWORD PTR __real@3ff8a3d70a3d70a4
        movsdx xmm0, QWORD PTR __real@40400147ae147ae1
        call    pow
        lea    rcx, OFFSET FLAT:$SG1354
        movaps xmm1, xmm0
        movd   rdx, xmm1
        call    printf
        xor    eax, eax
        add    rsp, 40
        ↳ 00000028H
        ret    0
main    ENDP
```

MOVSDX Intel [[Int13](#)] y AMD [[AMD13a](#)], MOVSD. (:[A.6.2 on page 1234](#)). MOVSDX.

pow() XMM0 y XMM1, XMM0. RDX para printf().? printf()?

Listing 27.6: Con optimización GCC 4.4.6 x64

```
.LC2:
    .string "32.01 ^ 1.54 = %lf\n"
main:
```

```

sub    rsp, 8
movsd xmm1, QWORD PTR .LC0[rip]
movsd xmm0, QWORD PTR .LC1[rip]
call   pow
; XMM0
mov    edi, OFFSET FLAT:.LC2
mov    eax, 1 ;
call   printf
xor    eax, eax
add    rsp, 8
ret
.LC0:
.long 171798692
.long 1073259479
.LC1:
.long 2920577761
.long 1077936455

```

GCC. printf() XMM0. EAX para printf() [Mit13].

27.3

```

#include <stdio.h>

double d_max (double a, double b)
{
    if (a>b)
        return a;

    return b;
};

int main()
{
    printf ("%f\n", d_max (1.2, 3.4));
    printf ("%f\n", d_max (5.6, -4));
}

```

27.3.1 x64

Listing 27.7: Con optimización MSVC 2012 x64

```

a$ = 8
b$ = 16
d_max PROC

```

```

comisd  xmm0, xmm1
ja      SHORT $LN2@d_max
movaps  xmm0, xmm1
$LN2@d_max:
fatret  0
d_max   ENDP

```

Con optimización MSVC .

COMISD «Compare Scalar Ordered Double-Precision Floating-Point Values and Set EFLAGS»..

Sin optimización MSVC :

Listing 27.8: MSVC 2012 x64

```

a$ = 8
b$ = 16
d_max PROC
    movsd x QWORD PTR [rsp+16], xmm1
    movsd x QWORD PTR [rsp+8], xmm0
    movsd x xmm0, QWORD PTR a$[rsp]
    comisd x xmm0, QWORD PTR b$[rsp]
    jbe   SHORT $LN1@d_max
    movsd x xmm0, QWORD PTR a$[rsp]
    jmp   SHORT $LN2@d_max
$LN1@d_max:
    movsd x xmm0, QWORD PTR b$[rsp]
$LN2@d_max:
    fatret 0
d_max  ENDP

```

GCC 4.4.6 MAXSD («Return Maximum Scalar Double-Precision Floating-Point Value»)!

Listing 27.9: Con optimización GCC 4.4.6 x64

```

d_max:
    maxsd  xmm0, xmm1
    ret

```

27.3.2 x86

Listing 27.10: Con optimización MSVC 2012 x86

```

_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_d_max PROC
    movsd  xmm0, QWORD PTR _a$[esp-4]
    comisd xmm0, QWORD PTR _b$[esp-4]
    jbe     SHORT $LN1@d_max
    fld    QWORD PTR _a$[esp-4]
    ret    0
$LN1@d_max:
    fld    QWORD PTR _b$[esp-4]
    ret    0
_d_max ENDP

```

a y b ST(0).

OllyDbg, COMISD CF y PF:

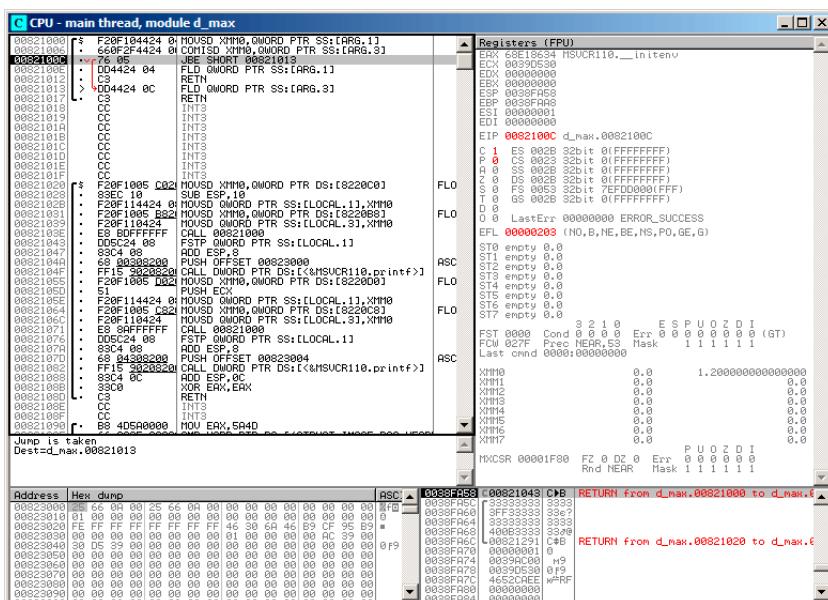


Figura 27.6: OllyDbg: COMISD CF y PF

27.4 :x64 y SIMD

:

Listing 27.11: Con optimización MSVC 2012 x64

```
v$ = 8
calculate_machine_epsilon PROC
    movsd QWORD PTR v$[rsp], xmm0
    movaps xmm1, xmm0
    inc QWORD PTR v$[rsp]
    movsd xmm0, QWORD PTR v$[rsp]
    subsd xmm0, xmm1
    ret 0
calculate_machine_epsilon ENDP
```

[2](#).

SUBSD «Subtract Scalar Double-Precision Floating-Point Values», ..

27.5

Listing 27.12: Con optimización MSVC 2012

```
__real@3f800000 DD 03f800000r ; 1
tv128 = -4
_tmp$ = -4
?float_rand@@YAMXZ PROC
    push    ecx
    call    ?my_rand@@YAIHZ
; EAX=
    and     eax, 8388607 ; 007✓
    ↴ fffffH
    or      eax, 1065353216 ; 3✓
    ↴ f800000H
; EAX= & 0x007fffff | 0x3f800000
;
    mov     DWORD PTR _tmp$[esp+4], eax
;
    movss  xmm0, DWORD PTR _tmp$[esp+4]
; 1.0:
    subss  xmm0, DWORD PTR __real@3f800000
;
```

[2](#).

```
    movss    DWORD PTR tv128[esp+4], xmm0
; ...
    fld      DWORD PTR tv128[esp+4]
    pop     ecx
    ret     0
?float_rand@@YAMXZ ENDP
```

27.6

-SD («Scalar Double-Precision»).

double float -SS («Scalar Single-Precision»), , MOVSS, COMISS, ADDSS, Spanish text placeholder.

«Scalar» .

Capítulo 28

Detalles específicos de ARM

28.1 (#) Signo numeral (#) antes del número

El compilador Keil, [IDA](#) y objdump anteponen a todos los números el signo «#», por ejemplo: Spanish text placeholder.[14.1.4](#). Pero cuando GCC 4.9 genera salida en lenguaje ensamblador, no lo hace, por ejemplo: Spanish text placeholder.[39.3](#).

Así que los listados para ARM en este libro están, en cierto sentido, mezclados.

Es difícil decir cuál es lo correcto. Probablemente conviene ceñirse a las convenciones aceptadas en el entorno en el que se trabaja.

28.2 Modos de direccionamiento

Esta instrucción es posible en ARM64:

```
ldr    x0, [x29,24]
```

Esto significa sumar 24 al valor en X29 y cargar el valor desde esa dirección. Ten en cuenta que 24 está dentro de los corchetes. El significado cambia si el número está fuera de los corchetes:

```
ldr    w4, [x1],28
```

Esto significa cargar el valor en la dirección contenida en X1 y luego sumar 28 a X1.

ARM permite sumar o restar una constante a/de la dirección usada para la carga. Y es posible hacerlo tanto antes como después de la carga.

Este modo de direccionamiento no existe en x86, pero sí en otros procesadores, incluso en el PDP-11. , *ptr++, *++ptr, *ptr--, *--ptr. Por cierto, esta es una característica de C difícil de recordar. Así es como funciona:

Término en C	Término en ARM	Expresión en C	cómo funciona
Post-incremento	post-indexed addressing	*ptr++	usar el valor de *ptr, luego increment el puntero
Post-decremento	post-indexed addressing	*ptr--	usar el valor de *ptr, luego decrement el puntero
Pre-incremento	pre-indexed addressing	*++ptr	increment el puntero y luego usar el valor de *
Pre-decremento	pre-indexed addressing	*--ptr	decrement el puntero y luego usar el valor de *

Pre-indexing se marca con un signo de exclamación en el lenguaje ensamblador de ARM. Por ejemplo, ver la línea 2 en Spanish text placeholder.[3.15](#).

Dennis Ritchie (uno de los creadores del lenguaje C) mencionó que probablemente fue idea de Ken Thompson (otro creador de C), porque esta característica del procesador ya estaba presente en el PDP-7 [Rit86][Rit93]. Por lo tanto, los compiladores de C pueden usarlo si está presente en el procesador de destino.

Todo esto es muy conveniente para trabajar con arreglos.

28.3 Cargar una constante en un registro

28.3.1 32 bits ARM

Como ya sabemos, todas las instrucciones tienen una longitud de 4 bytes en modo ARM y de 2 bytes en modo Thumb. Entonces, ¿cómo podemos cargar un valor de 32 bits en un registro si no es posible codificarlo en una sola instrucción?

Probemos:

```
unsigned int f()
{
    return 0x12345678;
}
```

Listing 28.1: GCC 4.6.3 -O3 Modo ARM

```
f:
    ldr      r0, .L2
    bx      lr
.L2:
    .word   305419896 ; 0x12345678
```

Es decir, el valor 0x12345678 se almacena aparte en memoria y se carga cuando es necesario. Pero también es posible evitar el acceso adicional a memoria.

Listing 28.2: GCC 4.6.3 -O3 -march=armv7-a (Modo ARM)

```
movw    r0, #22136      ; 0x5678
movt    r0, #4660       ; 0x1234
bx     lr
```

Se ve que el valor se carga en el registro por partes: primero la parte baja (con MOVW) y luego la alta (con MOVT).

Por lo tanto, se necesitan 2 instrucciones en modo ARM para cargar un valor de 32 bits en un registro. No es un gran problema, porque en el código real no hay tantas constantes (excepto 0 y 1). ¿Significa eso que la versión de dos instrucciones es más lenta que la de una sola instrucción? Probablemente no. Lo más seguro es que los procesadores ARM modernos detecten estas secuencias y las ejecuten rápidamente.

Por otro lado, [IDA](#) reconoce fácilmente estos patrones en el código y desensambla esta función como:

```
MOV    R0, 0x12345678
BX     LR
```

28.3.2 ARM64

```
uint64_t f()
{
    return 0x12345678ABCDEF01;
};
```

Listing 28.3: GCC 4.9.1 -O3

```
mov    x0, 61185      ; 0xef01
movk   x0, 0xabcd, lsl 16
movk   x0, 0x5678, lsl 32
movk   x0, 0x1234, lsl 48
ret
```

MOVK «MOV Keep», i.e., escribe un valor de 16 bits en el registro sin tocar el resto de los bits. LSL Por lo tanto, se necesitan 4 instrucciones para cargar un valor de 64 bits en un registro.

Almacenar un número en coma flotante en un registro

Algunos números se pueden escribir en un registro D usando solo una instrucción.

Por ejemplo:

```
double a()
{
    return 1.5;
}
```

Listing 28.4: GCC 4.9.1 -O3 + objdump

0000000000000000 <a>:	
0: 1e6f1000	fmov d0, #1.5000000000000000e+000
4: d65f03c0	ret

El número 1.5 efectivamente fue codificado en una instrucción de 32 bits. ¿Pero cómo? En ARM64, la instrucción FMOV dispone de 8 bits para codificar ciertos números en coma flotante. El algoritmo se denomina VFPEExpandImm() en [ARM13a]. Esto también se llama *minifloat*¹. Podemos probar con distintos valores: el compilador puede codificar 30.0 y 31.0, pero no pudo codificar 32.0, por lo que se deben reservar 8 bytes en memoria y escribirlo allí en formato IEEE 754:

```
double a()
{
    return 32;
}
```

Listing 28.5: GCC 4.9.1 -O3

a:	ldr d0, .LC0
	ret
.LC0:	.word 0
	.word 1077936128

28.4 en ARM64 Relocaciones en ARM64

Como sabemos, en ARM64 las instrucciones tienen 4 bytes, por lo que es imposible escribir un número grande en un registro con una sola instrucción. Sin embargo, una imagen ejecutable puede cargarse en cualquier dirección aleatoria de memoria;

¹[wikipedia](#)

para eso existen las reloaciones. Más sobre ellas (en relación con Win32 PE): [68.2.6](#) en [page 903](#).

En ARM64 se adopta el siguiente método: la dirección se forma mediante el par de instrucciones ADRP y ADD. La primera carga en un registro la dirección de una página de 4 KB y la segunda suma el resto. Compilaremos el ejemplo de «Hello, world!» (Spanish text placeholder.[6](#)) en GCC (Linaro) 4.9 en win32:

Listing 28.6: GCC (Linaro) 4.9 y objdump

```
...>aarch64-linux-gnu-gcc.exe hw.c -c
...>aarch64-linux-gnu-objdump.exe -d hw.o
...
0000000000000000 <main>:
 0:   a9bf7bfd      stp    x29, x30, [sp,#-16]!
 4:   910003fd      mov    x29, sp
 8:   90000000      adrp   x0, 0 <main>
 c:   91000000      add    x0, x0, #0x0
10:   94000000      bl     0 <printf>
14:   52800000      mov    w0, #0x0
  ↴ // #0
18:   a8c17bfd      ldp    x29, x30, [sp],#16
1c:   d65f03c0      ret
...
...>aarch64-linux-gnu-objdump.exe -r hw.o
...
RELOCATION RECORDS FOR [.text]:
OFFSET          TYPE           VALUE
0000000000000008 R_AARCH64_ADR_PREL_PG_HI21 .rodata
000000000000000c R_AARCH64_ADD_ABS_L012_NC .rodata
0000000000000010 R_AARCH64_CALL26   printf
```

Así que en este archivo objeto hay 3 reloaciones.

- La primera toma la dirección de la página, recorta los 12 bits inferiores y escribe los 21 bits altos restantes en los campos de bits de la instrucción ADRP. Esto se debe a que no es necesario codificar los 12 bits bajos y la instrucción ADRP solo tiene espacio para 21 bits.
- La segunda coloca los 12 bits de la dirección relativa al inicio de la página en los campos de bits de la instrucción ADD.
- La última, de 26 bits, se aplica a la instrucción en la dirección 0x10 donde está el salto a la función printf(). Todas las direcciones de instrucciones en

ARM64 (y en ARM en modo ARM) tienen ceros en los dos bits menos significativos (porque todas las instrucciones tienen un tamaño de 4 bytes), así que solo es necesario codificar los 26 bits superiores del espacio de direcciones de 28 bits (± 128 MB).

En el archivo ejecutable enlazado no hay tales relocalizaciones en estos lugares, porque ya se conoce dónde estará la cadena «Hello!», en qué página, y también se conoce la dirección de la función `puts()`. Por lo tanto, los valores ya están establecidos en las instrucciones `ADRP`, `ADD` y `BL` (el enlazador los escribió durante el enlace):

Listing 28.7: objdump

```
0000000000400590 <main>:
 400590:   a9bf7bfd      stp    x29, x30, [sp,#-16]!
 400594:   910003fd      mov    x29, sp
 400598:   90000000      adrp   x0, 400000 <_init-0x3b8>
  ↴ >
 40059c:   91192000      add    x0, x0, #0x648
 4005a0:   97fffffa0     bl     400420 <puts@plt>
 4005a4:   52800000      mov    w0, #0x0
  ↴       // #0
 4005a8:   a8c17bfd      ldp    x29, x30, [sp],#16
 4005ac:   d65f03c0      ret

...
Contents of section .rodata:
 400640 01000200 00000000 48656c6c 6f210000 .....Hello!..
```

Como ejemplo, intentemos desensamblar manualmente la instrucción `BL`.
`0x97fffffa0 10010111111111111111110100000b`. De acuerdo con [ARM13a, pág. C5.6.26], `imm26` son los últimos 26 bits: $imm26 = 111111111111111111110100000$. Es `0x3FFFFA0`, pero el MSB es 1, por lo que el número es negativo y, en términos de aritmética modular módulo 2^{32} , es `0xFFFFFFF80`. De nuevo, módulo 2^{32} , $0xFFFFFFF80 * 4 = 0xFFFFFE80$, y $0x4005a0 + FFFF80 = 0x400420$ (ten en cuenta: consideraremos la dirección de la instrucción `BL`, no el valor actual de `PC`, que puede ser distinto). Así que la dirección de destino es `0x400420`.

Más sobre relocalizaciones relacionadas con ARM64: [ARM13b].

Capítulo 29

Detalles específicos de MIPS

29.1 Carga de constantes en un registro

```
unsigned int f()
{
    return 0x12345678;
};
```

En MIPS, al igual que en ARM, todas las instrucciones tienen 32 bits, por lo que no es posible incrustar una constante de 32 bits en una sola instrucción. Por lo tanto, esto se traduce en al menos dos instrucciones: la primera carga la parte alta del número de 32 bits y la segunda aplica una operación OR, que en la práctica establece los 16 bits bajos del registro de destino.

Listing 29.1: GCC 4.4.5 -O3 (salida de assembly)

li	\$2,305397760	# 0x12340000
j	\$31	
ori	\$2,\$2,0x5678 ; branch delay slot	

IDA IDA conoce estos patrones de código frecuentes y, para mayor comodidad, muestra la última instrucción ORI como la pseudoinstrucción LI, que supuestamente carga un número completo de 32 bits en el registro \$V0.

Listing 29.2: GCC 4.4.5 -O3 (IDA)

lui	\$v0, 0x1234
jr	\$ra
li	\$v0, 0x12345678 ; branch delay slot

En la salida de ensamblador de GCC aparece la pseudoinstrucción LUI, pero en realidad se trata de LUI («Load Upper Immediate»), que escribe un valor de 16 bits en la parte alta del registro.

29.2 Lecturas adicionales sobre MIPS

[[Swe10](#)].

Parte II



Capítulo 30

01111111	0x7f	127	127
01111110	0x7e	126	126
...			
00000110	0x6	6	6
00000101	0x5	5	5
00000100	0x4	4	4
00000011	0x3	3	3
00000010	0x2	2	2
00000001	0x1	1	1
00000000	0x0	0	0
11111111	0xff	255	-1
11111110	0xfe	254	-2
11111101	0xfd	253	-3
11111100	0xfc	252	-4
11111011	0xfb	251	-5
11111010	0xfa	250	-6
...			
10000010	0x82	130	-126
10000001	0x81	129	-127
10000000	0x80	128	-128

La diferencia entre números con y sin signo es que si representamos 0xFFFFFFFF y 0x00000002 como sin signo, entonces el primer número (4294967294) es mayor que el segundo (2). Si los representamos ambos como con signo, el primero es -2, que es menor que el segundo (2). Esa es la razón por la que los saltos condicionales ([12 on page 146](#)) están presentes tanto para comparaciones con signo (ej. JG, JL) como sin signo (JA, JB).

:

- .
- :
 - `int64_t` (-9,223,372,036,854,775,808..9,223,372,036,854,775,807) (-9.2.. 9.2) o `0x8000000000000000..0x7FFFFFFFFFFFFFFF`,
 - `int` (-2,147,483,648..2,147,483,647) (- 2.15.. 2.15Gb) o `0x80000000..0x7FFFFFFF`,
 - `char` (-128..127 o `0x80..0x7F`),
 - `ssize_t`.
- :
 - `uint64_t` (0..18,446,744,073,709,551,615 (18) o `0..0xFFFFFFFFFFFFFFF`)
 - `unsigned int` (0..4,294,967,295 (4.3Gb) o `0..0xFFFFFFFF`),
 - `unsigned char` (0..255 o `0..0xFF`),
 - `size_t`.
- .
- [24.5 on page 531](#).
-
- . : IDIV/IMUL y DIV/MUL .
- : CBW/CWD/CWDE/CDQ/CDQE ([A.6.3 on page 1237](#)), MOVSX ([15.1.1 on page 246](#)), SAR ([A.6.3 on page 1242](#)).

Capítulo 31

Endianness

31.1 Big-endian

0x12345678 :

+0	0x12
+1	0x34
+2	0x56
+3	0x78

Motorola 68k, IBM POWER.

31.2 Little-endian

0x12345678 :

+0	0x78
+1	0x56
+2	0x34
+3	0x12

Intel x86.

31.3 Ejemplo

¹.

:

```
#include <stdio.h>

int main()
{
    int v, i;

    v=123;

    printf ("%02X %02X %02X %02X\n",
            *(char*)&v,
            *((char*)&v)+1,
            *((char*)&v)+2,
            *((char*)&v)+3);
}
```

:

```
root@debian-mips:~# ./a.out
00 00 00 7B
```

.0x7B 123.

«mips» (big-endian) y «mipsel» (little-endian)).

21.4.3 on page 472.

31.4 Bi-endian

ARM, PowerPC, SPARC, MIPS, IA64², Spanish text placeholder.

31.5

The BSWAP .

htonl() y htons().

«network byte order», «host byte order».

¹: <http://go.yurichev.com/17008>

²Intel Architecture 64 (Itanium): 93 on page 1156

Capítulo 32

- . AKA¹ «static memory allocation». : [7.2 on page 88](#).
- . AKA «allocate on stack». . : [18.2 on page 334](#).
- . AKA «dynamic memory allocation». malloc()/free() o new/delete en C++... : [21.2 on page 450](#).

¹Also Known As (También conocido como)

Capítulo 33

CPU

33.1

[.. : 12.1.2 on page 158](#), [12.3 on page 168](#), [19.5.2 on page 419](#).

33.2

Capítulo 34

• •

• • •

34.1 ?

```
4 6 0 1 3 5 7 8 9 2
```

:

• ;

• ;

• .

:

```
4 6 0 1 3 5 7 8 9 2
      ^   ^
```

:

```
4 6 0 1 7 5 3 8 9 2
```

Parte III

Capítulo 35

$$C = \frac{5 \cdot (F - 32)}{9}$$

:1); 2).

exit().

35.1

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int celsius, fahr;
    printf ("Enter temperature in Fahrenheit:\n");
    if (scanf ("%d", &fahr)!=1)
    {
        printf ("Error while parsing your input\n");
        exit(0);
    };

    celsius = 5 * (fahr-32) / 9;

    if (celsius<-273)
    {
        printf ("Error: incorrect temperature!\n");
        exit(0);
    };
    printf ("Celsius: %d\n", celsius);
```

};

35.1.1 Con optimización MSVC 2012 x86

Listing 35.1: Con optimización MSVC 2012 x86

```

$SG4228 DB      'Enter temperature in Fahrenheit:', 0aH, 00H
$SG4230 DB      '%d', 00H
$SG4231 DB      'Error while parsing your input', 0aH, 00H
$SG4233 DB      'Error: incorrect temperature!', 0aH, 00H
$SG4234 DB      'Celsius: %d', 0aH, 00H

_fahr$ = -4                                ; size ↴
↳ = 4
_main PROC
    push    ecx
    push    esi
    mov     esi, DWORD PTR __imp__printf
    push    OFFSET $SG4228          ; 'Enter temperature in'
    ↳ Fahrenheit:'
    call    esi                  ; printf()
    lea    eax, DWORD PTR _fahr$[esp+12]
    push    eax
    push    OFFSET $SG4230          ; '%d'
    call    DWORD PTR __imp__scanf
    add    esp, 12                ; ↴
    ↳ 0000000cH
    cmp    eax, 1
    je     SHORT $LN2@main
    push    OFFSET $SG4231          ; 'Error while parsing '
    ↳ your input'
    call    esi                  ; printf()
    add    esp, 4
    push    0
    call    DWORD PTR __imp__exit
$LN9@main:
$LN2@main:
    mov    eax, DWORD PTR _fahr$[esp+8]
    add    eax, -32                ; ↴
    ↳ ffffffe0H
    lea    ecx, DWORD PTR [eax+eax*4]
    mov    eax, 954437177          ; 38 ↴
    ↳ e38e39H
    imul   ecx
    sar    edx, 1
    mov    eax, edx

```

```

        shr    eax, 31 ; ↵
↳ 0000001fH
        add    eax, edx
        cmp    eax, -273 ; ↵
↳ ffffffeefH
        jge    SHORT $LN1@main
        push   OFFSET $SG4233 ; 'Error: incorrect ↵
↳ temperature!'
        call   esi ; printf()
        add    esp, 4
        push   0
        call   DWORD PTR __imp__exit

$LN10@main:
$LN1@main:
        push   eax
        push   OFFSET $SG4234 ; 'Celsius: %d'
        call   esi ; printf()
        add    esp, 8
        ;
        xor    eax, eax
        pop    esi
        pop    ecx
        ret    0

$LN8@main:
_main    ENDP

```

- printf() ESI printf() CALL ESI..
- ADD EAX, -32 . $EAX = EAX + (-32)$ $EAX = EAX - 32$ ADD SUB.
- LEA 5: lea ecx, DWORD PTR [eax+eax*4]., $i+i*4$ $i*5$ y LEA IMUL.
SHL EAX, 2 / ADD EAX, EAX.
- ([41 on page 631](#)).
- main() 0 return 0. [[ISO07](#), pág. 5.1.2.2.3] main() 0 return..

35.1.2 Con optimización MSVC 2012 x64

```

xor    ecx, ecx
call  QWORD PTR __imp__exit
int   3

```

INT 3.

```
exit() 1 .
```

35.2

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    double celsius, fahr;
    printf ("Enter temperature in Fahrenheit:\n");
    if (scanf ("%lf", &fahr)!=1)
    {
        printf ("Error while parsing your input\n");
        exit(0);
    };

    celsius = 5 * (fahr-32) / 9;

    if (celsius<-273)
    {
        printf ("Error: incorrect temperature!\n");
        exit(0);
    };
    printf ("Celsius: %lf\n", celsius);
}
```

MSVC 2010 x86 FPU...

Listing 35.2: Con optimización MSVC 2010 x86

... MSVC 2012 SIMD :

Listing 35.3: Con optimización MSVC 2010 x86

```
$SG4228 DB      'Enter temperature in Fahrenheit:', 0aH, 00H
$SG4230 DB      '%lf', 00H
$SG4231 DB      'Error while parsing your input', 0aH, 00H
$SG4233 DB      'Error: incorrect temperature!', 0aH, 00H
$SG4234 DB      'Celsius: %lf', 0aH, 00H
__real@c0711000000000000 DQ 0c0711000000000000r ; -273
__real@4040000000000000 DQ 0404000000000000r ; 32
__real@4022000000000000 DQ 0402200000000000r ; 9
```

¹longjmp()

```

__real@4014000000000000 DQ 0401400000000000r ; 5

_fahr$ = -8 ; size ↴
    ↵ = 8
_main PROC
    sub    esp, 8
    push   esi
    mov    esi, DWORD PTR __imp__printf
    push   OFFSET $SG4228 ; 'Enter temperature in ↴
    ↵ Farenheit:'
    call   esi ; printf()
    lea    eax, DWORD PTR _fahr$[esp+16]
    push   eax
    push   OFFSET $SG4230 ; '%lf'
    call   DWORD PTR __imp__scanf
    add    esp, 12 ; ↴
    ↵ 0000000cH
    cmp    eax, 1
    je     SHORT $LN2@main
    push   OFFSET $SG4231 ; 'Error while parsing ↴
    ↵ your input'
    call   esi ; printf()
    add    esp, 4
    push   0
    call   DWORD PTR __imp__exit

$LN9@main:
$LN2@main:
    movsd xmm1, QWORD PTR _fahr$[esp+12]
    subsd xmm1, QWORD PTR __real@4040000000000000 ; 32
    movsd xmm0, QWORD PTR __real@c071100000000000 ; -273
    mulsd xmm1, QWORD PTR __real@4014000000000000 ; 5
    divsd xmm1, QWORD PTR __real@4022000000000000 ; 9
    comisd xmm0, xmm1
    jbe    SHORT $LN1@main
    push   OFFSET $SG4233 ; 'Error: incorrect ↴
    ↵ temperature!'
    call   esi ; printf()
    add    esp, 4
    push   0
    call   DWORD PTR __imp__exit

$LN10@main:
$LN1@main:
    sub    esp, 8
    movsd QWORD PTR [esp], xmm1
    push   OFFSET $SG4234 ; 'Celsius: %lf'
    call   esi ; printf()
    add    esp, 12 ; ↴

```

```
↳ 0000000cH
;
xor    eax, eax
pop    esi
add    esp, 8
ret    0
$LN8@main:
_main  ENDP
```

, SIMD..

-273 XMM0 ..

Capítulo 36

¹...

:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181...

36.1 #1

21.

```
#include <stdio.h>

void fib (int a, int b, int limit)
{
    printf ("%d\n", a+b);
    if (a+b > limit)
        return;
    fib (b, a+b, limit);
}

int main()
{
    printf ("0\n1\n1\n");
    fib (1, 1, 20);
}
```

Listing 36.1: MSVC 2010 x86

```
_TEXT  SEGMENT
_a$ = 8           ; size = 4
_b$ = 12          ; size = 4
```

¹<http://go.yurichev.com/17332>

```
_limit$ = 16      ; size = 4
_fib    PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    add     eax, DWORD PTR _b$[ebp]
    push    eax
    push    OFFSET $SG2750 ; "%d"
    call    DWORD PTR __imp__printf
    add     esp, 8
    mov     ecx, DWORD PTR _limit$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp]
    add     edx, DWORD PTR _b$[ebp]
    push    edx
    mov     eax, DWORD PTR _b$[ebp]
    push    eax
    call    _fib
    add     esp, 12
    pop    ebp
    ret    0
_fib    ENDP

_main   PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG2753 ; "0\n1\n1\n"
    call    DWORD PTR __imp__printf
    add     esp, 4
    push    20
    push    1
    push    1
    call    _fib
    add     esp, 12
    xor     eax, eax
    pop    ebp
    ret    0
_main   ENDP
```

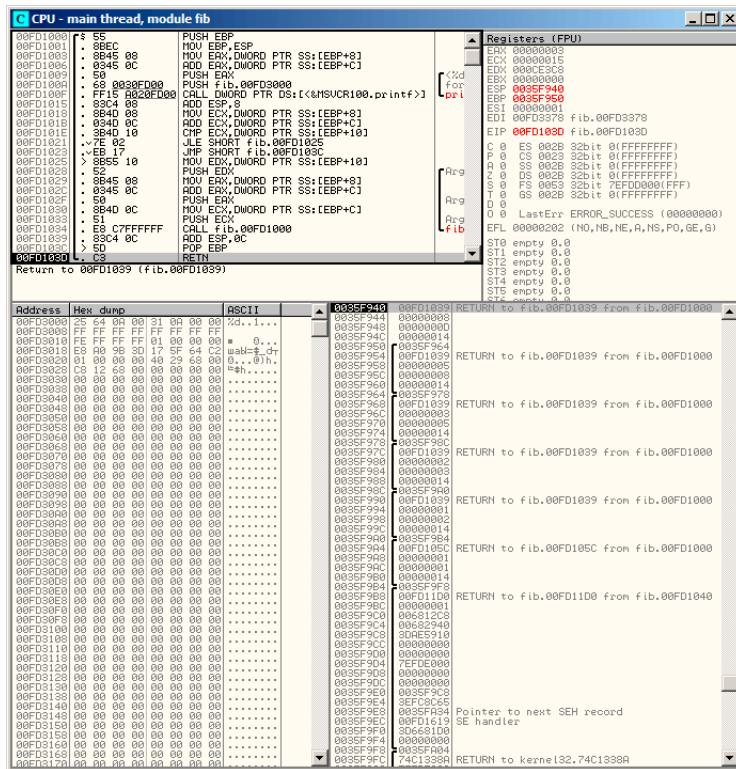


Figura 36.1: OllyDbg:

2.

```

0035F940 00FD1039 RETURN to fib.00FD1039 from fib.00FD1000
0035F944 00000008 : a
0035F948 0000000D : b
0035F94C 00000014 : limit
0035F950 /0035F964
0035F954 |00FD1039 RETURN to fib.00FD1039 from fib.00FD1000
0035F958 |00000005 : a
0035F95C |00000008 : b
0035F960 |00000014 : limit
0035F964 ]0035F978
0035F968 |00FD1039 RETURN to fib.00FD1039 from fib.00FD1000
0035F96C |00000003 : a
0035F970 |00000005 : b
0035F974 |00000014 : limit
0035F978 ]0035F98C
0035F97C |00FD1039 RETURN to fib.00FD1039 from fib.00FD1000
0035F980 |00000002 : a
0035F984 |00000003 : b
0035F988 |00000014 : limit
0035F98C ]0035F9A0
0035F990 |00FD1039 RETURN to fib.00FD1039 from fib.00FD1000
0035F994 |00000001 : a
0035F998 |00000002 : b
0035F99C |00000014 : limit
0035F9A0 ]0035F9B4
0035F9A4 |00FD105C RETURN to fib.00FD105C from fib.00FD1000
0035F9A8 |00000001 : a \
0035F9AC |00000001 : b | f1()
0035F9B0 |00000014 : limit /
0035F9B4 ]0035F9F8
0035F9B8 |00FD11D0 RETURN to fib.00FD11D0 from fib.00FD1040
0035F9BC |00000001 main() : argc \
0035F9C0 |006812C8 main() : argv | main()
0035F9C4 |00682940 main() : envp /

```

³, . limit (0x14 o 20), a y b . . OllyDbg . . .

main().argc 1 ()

0xC00000FD ()

```
:
#include <stdio.h>

void fib (int a, int b, int limit)
{
    int next=a+b;
    printf ("%d\n", next);
    if (next > limit)
        return;
    fib (b, next, limit);
};

int main()
{
    printf ("0\n1\n1\n");
    fib (1, 1, 20);
}
```

```
:
```

Listing 36.2: MSVC 2010 x86

```
_next$ = -4      ; size = 4
_a$ = 8         ; size = 4
_b$ = 12        ; size = 4
_limit$ = 16    ; size = 4
_fib     PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]
    add     eax, DWORD PTR _b$[ebp]
    mov     DWORD PTR _next$[ebp], eax
    mov     ecx, DWORD PTR _next$[ebp]
    push    ecx
    push    OFFSET $SG2751 ; '%d'
    call   DWORD PTR __imp__printf
    add     esp, 8
    mov     edx, DWORD PTR _next$[ebp]
    cmp     edx, DWORD PTR _limit$[ebp]
    jle     SHORT $LN1@fib
    jmp     SHORT $LN2@fib
$LN1@fib:
    mov     eax, DWORD PTR _limit$[ebp]
    push    eax
```

```
        mov     ecx, DWORD PTR _next$[ebp]
        push    ecx
        mov     edx, DWORD PTR _b$[ebp]
        push    edx
        call    _fib
        add    esp, 12
$LN2@fib:
        mov     esp, ebp
        pop    ebp
        ret    0
_fib    ENDP

_main   PROC
        push    ebp
        mov     ebp, esp
        push    OFFSET $SG2753 ; "0\n1\n1\n"
        call    DWORD PTR __imp__printf
        add    esp, 4
        push    20
        push    1
        push    1
        call    _fib
        add    esp, 12
        xor    eax, eax
        pop    ebp
        ret    0
_main   ENDP
```

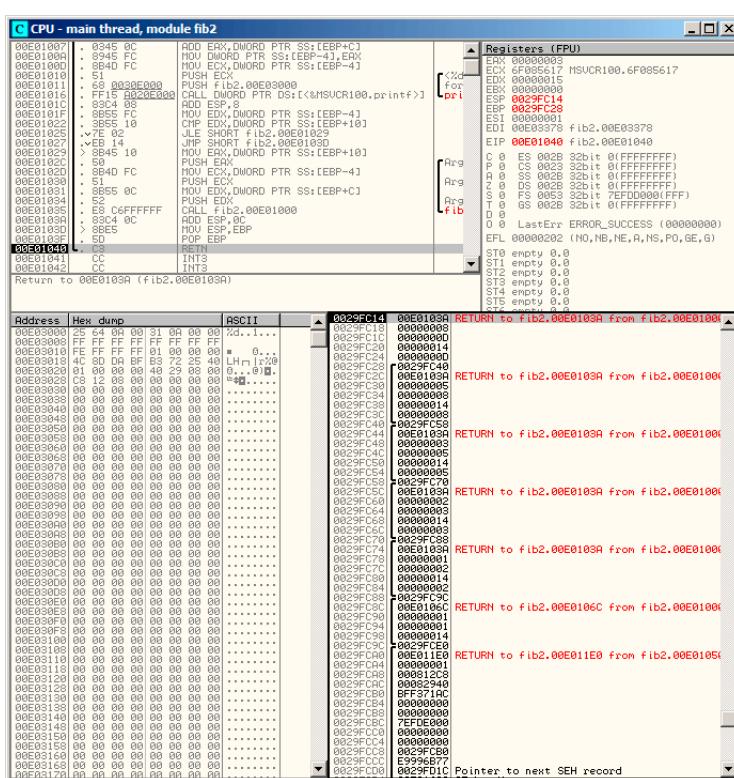


Figura 36.2: OllyDbg:

:

```

0029FC14 00E0103A RETURN to fib2.00E0103A from fib2.00E01000
0029FC18 00000008 : a
0029FC1C 0000000D : b
0029FC20 00000014 : limit
0029FC24 0000000D
0029FC28 /0029FC40
0029FC2C |00E0103A RETURN to fib2.00E0103A from fib2.00E01000
0029FC30 |00000005 : a
0029FC34 |00000008 : b
0029FC38 |00000014 : limit
0029FC3C |00000008
0029FC40 ]0029FC58
0029FC44 |00E0103A RETURN to fib2.00E0103A from fib2.00E01000
0029FC48 |00000003 : a
0029FC4C |00000005 : b
0029FC50 |00000014 : limit
0029FC54 |00000005
0029FC58 ]0029FC70
0029FC5C |00E0103A RETURN to fib2.00E0103A from fib2.00E01000
0029FC60 |00000002 : a
0029FC64 |00000003 : b
0029FC68 |00000014 : limit
0029FC6C |00000003
0029FC70 ]0029FC88
0029FC74 |00E0103A RETURN to fib2.00E0103A from fib2.00E01000
0029FC78 |00000001 : a           \
0029FC7C |00000002 : b           | prepared in f1() for next \
    ↳ f1()
0029FC80 |00000014 : limit      /
0029FC84 |00000002
0029FC88 ]0029FC9C
0029FC8C |00E0106C RETURN to fib2.00E0106C from fib2.00E01000
0029FC90 |00000001 : a           \
0029FC94 |00000001 : b           | prepared in main() for f1 \
    ↳ ()
0029FC98 |00000014 : limit      /
0029FC9C ]0029FCE0
0029FCA0 |00E011E0 RETURN to fib2.00E011E0 from fib2.00E01050
0029FCA4 |00000001 main() : argc \
0029FCA8 |000812C8 main() : argv | main()
0029FCAC |00082940 main() : envp /

```


Capítulo 37

```
/* By Bob Jenkins, (c) 2006, Public Domain */

#include <stdio.h>
#include <stddef.h>
#include <string.h>

typedef unsigned long ub4;
typedef unsigned char ub1;

static const ub4 crctab[256] = {
    0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x2
    ↴ x706af48f,
    0xe963a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x2
    ↴ x97d2d988,
    0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064, 0x2
    ↴ x6ab020f2,
    0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x2
    ↴ x83d385c7,
    0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec, 0x14015c4f, 0x2
    ↴ x63066cd9,
    0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0x2
    ↴ xa2677172,
    0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x2
    ↴ x42b2986c,
    0xdbbbc9d6, 0xacbcf940, 0x32d86ce3, 0x45df5c75, 0xdc60dcf, 0x2
    ↴ xabd13d59,
    0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116, 0x21b4f4b5, 0x2
    ↴ x56b3c423,
    0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0x2
    ↴ xb10be924,
    0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x2
    ↴ x01db7106,
```

0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0x
 ↳ xe8b8d433,
 0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x
 ↳ x086d3d2d,
 0x91646c97, 0xe6635c01, 0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0x
 ↳ xf262004e,
 0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x
 ↳ x12b7e950,
 0x8bbeb8ea, 0xfc9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0x
 ↳ xfb44c65,
 0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0x4adfa541, 0x
 ↳ x3dd895d7,
 0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846, 0x
 ↳ xda60b8d0,
 0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x
 ↳ x270241aa,
 0xbe0b1010, 0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0x
 ↳ xce61e49f,
 0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17, 0x
 ↳ x2eb40d81,
 0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x
 ↳ x74b1d29a,
 0xeadd54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x
 ↳ x94643b84,
 0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x
 ↳ x7d079eb1,
 0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x
 ↳ x806567cb,
 0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x
 ↳ x67dd4acc,
 0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5, 0xd6d6a3e8, 0x
 ↳ xa1d1937e,
 0x38d8c2c4, 0x4fdfff252, 0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x
 ↳ x48b2364b,
 0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60, 0xdf60efc3, 0x
 ↳ xa867df55,
 0x316e8eef, 0x4669be79, 0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x
 ↳ x5268e236,
 0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe, 0x
 ↳ xb2bd0b28,
 0x2bb45a92, 0x5cb36a04, 0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x
 ↳ x5bdeae1d,
 0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a, 0x9c0906a9, 0x
 ↳ xeb0e363f,
 0x72076785, 0x05005713, 0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x
 ↳ x0cb61b38,

```

0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0xbdbdf21, 0x86d3d2d4, 0x
    ↴ xf1d4e242,
0x68ddb3f8, 0x1fda836e, 0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x
    ↴ x18b74777,
0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c, 0x8f659eff, 0x
    ↴ xf862ae69,
0x616bfdf3, 0x166ccf45, 0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x
    ↴ x3903b3c2,
0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db, 0xaea16a4a, 0x
    ↴ xd9d65adc,
0x40df0b66, 0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x
    ↴ x30b5ffe9,
0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6, 0xbad03605, 0x
    ↴ xcdd70693,
0x54de5729, 0x23d967bf, 0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x
    ↴ x2a6f2b94,
0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d,
};

/* how to derive the values in crctab[] from polynomial 0x
   ↴ xedb88320 */
void build_table()
{
    ub4 i, j;
    for (i=0; i<256; ++i) {
        j = i;
        j = (j>>1) ^ ((j&1) ? 0xedb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        j = (j>>1) ^ ((j&1) ? 0edb88320 : 0);
        printf("0x%8lx, ", j);
        if (i%6 == 5) printf("\n");
    }
}

/* the hash function */
ub4 crc(const void *key, ub4 len, ub4 hash)
{
    ub4 i;
    const ub1 *k = key;
    for (hash=len, i=0; i<len; ++i)
        hash = (hash >> 8) ^ crctab[(hash & 0xff) ^ k[i]];
}

```

```

    return hash;
}

/* To use, try "gcc -O crc.c -o crc; crc < crc.c" */
int main()
{
    char s[1000];
    while (gets(s)) printf("%.8lx\n", crc(s, strlen(s), 0));
    return 0;
}

```

```

_key$ = 8           ; size = 4
_len$ = 12          ; size = 4
_hash$ = 16          ; size = 4
_crc    PROC
    mov     edx, DWORD PTR _len$[esp-4]
    xor     ecx, ecx ;
    mov     eax, edx
    test    edx, edx
    jbe    SHORT $LN1@crc
    push    ebx
    push    esi
    mov     esi, DWORD PTR _key$[esp+4] ; ESI = key
    push    edi
$LL3@crc:
;
    movzx   edi, BYTE PTR [ecx+esi]
    mov     ebx, eax ; EBX = (hash = len)
    and     ebx, 255 ; EBX = hash & 0xff

; XOR EDI, EBX (EDI=EDI^EBX) -
;
;
;

    xor     edi, ebx

; EAX=EAX>>8;
    shr     eax, 8

; EAX=EAX^crctab[EDI*4] -
    xor     eax, DWORD PTR _crctab[edi*4]
    inc     ecx          ; i++
    cmp     ecx, edx      ; i<len ?

```

```

jb    SHORT $LL3@crc ;
pop   edi
pop   esi
pop   ebx
$LN1@crc:
ret   0
_crc  ENDP

```

```

public crc
proc near

key      = dword ptr  8
hash     = dword ptr  0Ch

push    ebp
xor    edx, edx
mov    ebp, esp
push    esi
mov    esi, [ebp+key]
push    ebx
mov    ebx, [ebp+hash]
test   ebx, ebx
mov    eax, ebx
jz    short loc_80484D3
nop
; 
lea    esi, [esi+0] ; 

loc_80484B8:
mov    ecx, eax      ;
xor    al, [esi+edx] ; AL=*(key+i)
add    edx, 1        ; i++
shr    ecx, 8        ; ECX=hash>>8
movzx  eax, al        ; EAX=*(key+i)
mov    eax, dword ptr ds:crctab[eax*4] ; EAX=<
        crctab[EAX]
        xor    eax, ecx      ; hash=EAX^ECX
        cmp    ebx, edx
        ja    short loc_80484B8

loc_80484D3:
pop    ebx
pop    esi
pop    ebp
retn
crc
\
```

Capítulo 38

/30	4	2	255.255.255.252	fffffffffc
/29	8	6	255.255.255.248	ffffffffff8
/28	16	14	255.255.255.240	ffffffffff0
/27	32	30	255.255.255.224	ffffffffe0
/26	64	62	255.255.255.192	fffffffc0
/24	256	254	255.255.255.0	fffffff00
/23	512	510	255.255.254.0	ffffffe00
/22	1024	1022	255.255.252.0	fffffc00
/21	2048	2046	255.255.248.0	fffff800
/20	4096	4094	255.255.240.0	fffff000
/19	8192	8190	255.255.224.0	fffffe000
/18	16384	16382	255.255.192.0	fffffc000
/17	32768	32766	255.255.128.0	fffff8000
/16	65536	65534	255.255.0.0	fffff0000
/8	16777216	16777214	255.0.0.0	ff000000

```
#include <stdio.h>
#include <stdint.h>

uint32_t form_IP (uint8_t ip1, uint8_t ip2, uint8_t ip3, ↴
                  uint8_t ip4)
{
    return (ip1<<24) | (ip2<<16) | (ip3<<8) | ip4;
};

void print_as_IP (uint32_t a)
{
    printf ("%d.%d.%d.%d\n",
            (a>>24)&0xFF,
            (a>>16)&0xFF,
            (a>>8)&0xFF,
```

```
        (a)&0xFF);
};

// bit=31..0
uint32_t set_bit (uint32_t input, int bit)
{
    return input=input|(1<<bit);
};

uint32_t form_netmask (uint8_t netmask_bits)
{
    uint32_t netmask=0;
    uint8_t i;

    for (i=0; i<netmask_bits; i++)
        netmask=set_bit(netmask, 31-i);

    return netmask;
};

void calc_network_address (uint8_t ip1, uint8_t ip2, uint8_t ↴
    ↳ ip3, uint8_t ip4, uint8_t netmask_bits)
{
    uint32_t netmask=form_netmask(netmask_bits);
    uint32_t ip=form_IP(ip1, ip2, ip3, ip4);
    uint32_t netw_adr;

    printf ("netmask=");
    print_as_IP (netmask);

    netw_adr=ip&netmask;

    printf ("network address=");
    print_as_IP (netw_adr);
};

int main()
{
    calc_network_address (10, 1, 2, 4, 24);      // 10.1.2.4, ↴
    ↳ /24
    calc_network_address (10, 1, 2, 4, 8);       // 10.1.2.4, ↴
    ↳ /8
    calc_network_address (10, 1, 2, 4, 25);      // 10.1.2.4, ↴
    ↳ /25
    calc_network_address (10, 1, 2, 64, 26);     // 10.1.2.4, ↴
    ↳ /26
};
```

38.1 calc_network_address()

`calc_network_address():`

Listing 38.1: Con optimización MSVC 2012 /Ob0

```

1 _ip1$ = 8           ; size = 1
2 _ip2$ = 12          ; size = 1
3 _ip3$ = 16          ; size = 1
4 _ip4$ = 20          ; size = 1
5 _netmask_bits$ = 24 ; size = 1
6 _calc_network_address PROC
7     push    edi
8     push    DWORD PTR _netmask_bits$[esp]
9     call    _form_netmask
10    push   OFFSET $SG3045 ; 'netmask='
11    mov     edi, eax
12    call   DWORD PTR __imp__printf
13    push   edi
14    call   _print_as_IP
15    push   OFFSET $SG3046 ; 'network address='
16    call   DWORD PTR __imp__printf
17    push   DWORD PTR _ip4$[esp+16]
18    push   DWORD PTR _ip3$[esp+20]
19    push   DWORD PTR _ip2$[esp+24]
20    push   DWORD PTR _ip1$[esp+28]
21    call   _form_IP
22    and    eax, edi      ; network address = host ↴
   ↳ address & netmask
23    push   eax
24    call   _print_as_IP
25    add    esp, 36
26    pop    edi
27    ret    0
28 _calc_network_address ENDP

```

38.2 form_IP()

`form_IP().`

:

-
-
- .

-
-

Listing 38.2: Sin optimización MSVC 2012

```

; ip1 "dd", ip2 "cc", ip3 "bb", ip4 "aa".
_ip1$ = 8           ; size = 1
_ip2$ = 12          ; size = 1
_ip3$ = 16          ; size = 1
_ip4$ = 20          ; size = 1
_form_IP PROC
    push    ebp
    mov     ebp, esp
    movzx   eax, BYTE PTR _ip1$[ebp]
    ; EAX=000000dd
    shl    eax, 24
    ; EAX=dd000000
    movzx   ecx, BYTE PTR _ip2$[ebp]
    ; ECX=000000cc
    shl    ecx, 16
    ; ECX=00cc0000
    or     eax, ecx
    ; EAX=ddcc0000
    movzx   edx, BYTE PTR _ip3$[ebp]
    ; EDX=000000bb
    shl    edx, 8
    ; EDX=0000bb00
    or     eax, edx
    ; EAX=ddccb00
    movzx   ecx, BYTE PTR _ip4$[ebp]
    ; ECX=000000aa
    or     eax, ecx
    ; EAX=ddccbbaa
    pop    ebp
    ret    0
_form_IP ENDP

```

Con optimización MSVC 2012 :

Listing 38.3: Con optimización MSVC 2012 /Ob0

```

; ip1 "dd", ip2 "cc", ip3 "bb", ip4 "aa".
_ip1$ = 8           ; size = 1
_ip2$ = 12          ; size = 1
_ip3$ = 16          ; size = 1
_ip4$ = 20          ; size = 1

```

```

_form_IP PROC
    movzx   eax, BYTE PTR _ip1$[esp-4]
    ; EAX=000000dd
    movzx   ecx, BYTE PTR _ip2$[esp-4]
    ; ECX=000000cc
    shl     eax, 8
    ; EAX=0000dd00
    or      eax, ecx
    ; EAX=0000ddcc
    movzx   ecx, BYTE PTR _ip3$[esp-4]
    ; ECX=000000bb
    shl     eax, 8
    ; EAX=00ddcc00
    or      eax, ecx
    ; EAX=00ddccbb
    movzx   ecx, BYTE PTR _ip4$[esp-4]
    ; ECX=000000aa
    shl     eax, 8
    ; EAX=ddccbb00
    or      eax, ecx
    ; EAX=ddccbbaa
    ret     0
_form_IP ENDP

```

Repetir 4 veces, para cada byte de entrada.

!

38.3 print_as_IP()

`print_as_IP()`

Listing 38.4: Sin optimización MSVC 2012

```

_a$ = 8                      ; size = 4
_print_as_IP PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    ; EAX=ddccbbaa
    and     eax, 255
    ; EAX=000000aa
    push    eax
    mov     ecx, DWORD PTR _a$[ebp]
    ; ECX=ddccbbaa

```

```

    shr      ecx, 8
    ; ECX=00ddccbb
    and      ecx, 255
    ; ECX=000000bb
    push     ecx
    mov      edx, DWORD PTR _a$[ebp]
    ; EDX=ddccbbaa
    shr      edx, 16
    ; EDX=0000ddcc
    and      edx, 255
    ; EDX=000000cc
    push     edx
    mov      eax, DWORD PTR _a$[ebp]
    ; EAX=ddccbbaa
    shr      eax, 24
    ; EAX=000000dd
    and      eax, 255 ;
    ; EAX=000000dd
    push     eax
    push     OFFSET $SG2973 ; '%d.%d.%d.%d'
    call    DWORD PTR __imp__printf
    add     esp, 20
    pop     ebp
    ret     0
_print_as_IP ENDP

```

Con optimización MSVC 2012

Listing 38.5: Con optimización MSVC 2012 /Ob0

```

_a$ = 8          ; size = 4
_print_as_IP PROC
    mov      ecx, DWORD PTR _a$[esp-4]
    ; ECX=ddccbbaa
    movzx   eax, cl
    ; EAX=000000aa
    push    eax
    mov      eax, ecx
    ; EAX=ddccbbaa
    shr      eax, 8
    ; EAX=00ddccbb
    and      eax, 255
    ; EAX=000000bb
    push    eax
    mov      eax, ecx
    ; EAX=ddccbbaa
    shr      eax, 16
    ; EAX=0000ddcc

```

```

and      eax, 255
; EAX=000000cc
push    eax
; ECX=ddccbbaa
shr     ecx, 24
; ECX=000000dd
push    ecx
push    OFFSET $SG3020 ; '%d.%d.%d.%d'
call    DWORD PTR __imp__printf
add    esp, 20
ret    0
_print_as_IP ENDP

```

38.4 form_netmask() y set_bit()

form_netmask(). set_bit().

Listing 38.6: Con optimización MSVC 2012 /Ob0

```

_input$ = 8          ; size = 4
_bit$ = 12         ; size = 4
_set_bit PROC
    mov    ecx, DWORD PTR _bit$[esp-4]
    mov    eax, 1
    shl    eax, cl
    or     eax, DWORD PTR _input$[esp-4]
    ret    0
_set_bit ENDP

_netmask_bits$ = 8      ; size = 1
_form_netmask PROC
    push   ebx
    push   esi
    movzx  esi, BYTE PTR _netmask_bits$[esp+4]
    xor    ecx, ecx
    xor    bl, bl
    test   esi, esi
    jle    SHORT $LN9@form_netma
    xor    edx, edx
$LL3@form_netma:
    mov    eax, 31
    sub    eax, edx
    push   eax
    push   ecx
    call   _set_bit
    inc    bl

```

```
    movzx   edx, bl
    add    esp, 8
    mov    ecx, eax
    cmp    edx, esi
    j1     SHORT $LL3@form_netma
$LN9@form_netma:
    pop    esi
    mov    eax, ecx
    pop    ebx
    ret    0
_form_netmask ENDP
```

set_bit().form_netmask()

38.5

!:

```
netmask=255.255.255.0
network address=10.1.2.0
netmask=255.0.0.0
network address=10.0.0.0
netmask=255.255.255.128
network address=10.1.2.0
netmask=255.255.255.192
network address=10.1.2.64
```

Capítulo 39

:

```
#include <stdio.h>

void f(int *a1, int *a2, size_t cnt)
{
    size_t i;

    //
    for (i=0; i<cnt; i++)
        a1[i*3]=a2[i*7];
}
```

?

39.1

Listing 39.1: Con optimización MSVC 2013 x64

```
f      PROC
; RDX=a1
; RCX=a2
; R8=cnt
    test    r8, r8          ; cnt==0?
    je     SHORT $LN1@f
    npad   11
$LL3@f:
    mov     eax, DWORD PTR [rdx]
    lea     rcx, QWORD PTR [rcx+12]
    lea     rdx, QWORD PTR [rdx+28]
```

```

        mov      DWORD PTR [rcx-12], eax
        dec      r8
        jne      SHORT $LL3@f
$LN1@f:
        ret      0
f      ENDP

```

C/C++:

```

#include <stdio.h>

void f(int *a1, int *a2, size_t cnt)
{
    size_t i;
    size_t idx1=0; idx2=0;

    //
    for (i=0; i<cnt; i++)
    {
        a1[idx1]=a2[idx2];
        idx1+=3;
        idx2+=7;
    };
}

```

39.2

Listing 39.2: Con optimización GCC 4.9 x64

```

; RDI=a1
; RSI=a2
; RDX=cnt
f:
    test    rdx, rdx ; cnt==0?
    je     .L1
;
    lea     rax, [0+rdx*4]
; RAX=RDX*4=cnt*4
    sal     rdx, 5
; RDX=RDX<<5=cnt*32
    sub     rdx, rax
; RDX=RDX-RAX=cnt*32-cnt*4=cnt*28
    add     rdx, rsi
; RDX=RDX+RSI=a2+cnt*28

```

```
.L3:
    mov      eax, DWORD PTR [rsi]
    add      rsi, 28
    add      rdi, 12
    mov      DWORD PTR [rdi-12], eax
    cmp      rsi, rdx
    jne      .L3
.L1:
    rep ret
```

[16.1.3 on page 262.](#)

```
#include <stdio.h>

void f(int *a1, int *a2, size_t cnt)
{
    size_t i;
    size_t idx1=0; idx2=0;
    size_t last_idx2=cnt*7;

    //
    for (;;)
    {
        a1[idx1]=a2[idx2];
        idx1+=3;
        idx2+=7;
        if (idx2==last_idx2)
            break;
    };
}
```

GCC (Linaro) 4.9 para ARM64 :

Listing 39.3: Con optimización GCC (Linaro) 4.9 ARM64

```
; X0=a1
; X1=a2
; X2=cnt
f:
    cbz      x2, .L1           ; cnt==0?
;
    add      x2, x2, x2, lsl 1
; X2=X2+X2<<1=X2+X2*2=X2*3
    mov      x3, 0
    lsl      x2, x2, 2
; X2=X2<<2=X2*4=X2*3*4=X2*12
```

```
.L3:
    ldr    w4, [x1],28      ;
    str    w4, [x0,x3]      ; X0+X3=a1+X3
    add    x3, x3, 12       ; X3
    cmp    x3, x2           ; ?
    bne    .L3

.L1:
    ret
```

GCC 4.4.5 para MIPS :

Listing 39.4: Con optimización GCC 4.4.5 para MIPS (IDA)

```
; $a0=a1
; $a1=a2
; $a2=cnt
f:
; :
        beqz    $a2, locret_24
; :
        move    $v0, $zero ; branch delay slot, NOP

loc_8:
; $a1
        lw      $a3, 0($a1)
; (i):
        addiu   $v0, 1
; :
        sltu    $v1, $v0, $a2
; $a0:
        sw      $a3, 0($a0)
; 0x1C (28) \$a1 :
        addiu   $a1, 0x1C
; i<cnt:
        bnez    $v1, loc_8
; 0xC (12) \$a0 :
        addiu   $a0, 0xC ; branch delay slot

locret_24:
        jr     $ra
        or     $at, $zero ; branch delay slot, NOP
```

39.3 Intel C++ 2011

:

Listing 39.5: Con optimización Intel C++ 2011 x64

```

f      PROC
; parameter 1: rcx = a1
; parameter 2: rdx = a2
; parameter 3: r8   = cnt
.B1.1::                      ; Preds .B1.0
    test    r8, r8
    ↴ ;8.14
    jbe     exit       ; Prob 50%
    ↴ ;8.14
                                ; LOE rdx rcx rbx rbp rsi rdi ✗
    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✗
    ↴ xmm13 xmm14 xmm15
.B1.2::                      ; Preds .B1.1
    cmp     r8, 6
    ↴ ;8.2
    jbe     just_copy    ; Prob 50%
    ↴ ;8.2
                                ; LOE rdx rcx rbx rbp rsi rdi ✗
    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✗
    ↴ xmm13 xmm14 xmm15
.B1.3::                      ; Preds .B1.2
    cmp     rcx, rdx
    ↴ ;9.11
    jbe     .B1.5       ; Prob 50%
    ↴ ;9.11
                                ; LOE rdx rcx rbx rbp rsi rdi ✗
    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✗
    ↴ xmm13 xmm14 xmm15
.B1.4::                      ; Preds .B1.3
    mov     r10, r8
    ↴ ;9.11
    mov     r9, rcx
    ↴ ;9.11
    shl     r10, 5
    ↴ ;9.11
    lea     rax, QWORD PTR [r8*4]
    ↴ ;9.11
    sub     r9, rdx
    ↴ ;9.11
    sub     r10, rax
    ↴ ;9.11
    cmp     r9, r10
    ↴ ;9.11
    jge     just_copy2   ; Prob 50%
    ↴ ;9.11
                                ; LOE rdx rcx rbx rbp rsi rdi ✗

```

```

    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✓
    ↴ xmm13 xmm14 xmm15
.B1.5:: ; Preds .B1.3 .B1.4
    cmp      rdx, rcx ✓
    ↴ ;9.11
    jbe     just_copy ; Prob 50% ✓
    ↴ ;9.11 ; LOE rdx rcx rbx rbp rsi rdi ✓
    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✓
    ↴ xmm13 xmm14 xmm15
.B1.6:: ; Preds .B1.5
    mov      r9, rdx ✓
    ↴ ;9.11
    lea      rax, QWORD PTR [r8*8] ✓
    ↴ ;9.11
    sub      r9, rcx ✓
    ↴ ;9.11
    lea      r10, QWORD PTR [rax+r8*4] ✓
    ↴ ;9.11
    cmp      r9, r10 ✓
    ↴ ;9.11
    jl     just_copy ; Prob 50% ✓
    ↴ ;9.11 ; LOE rdx rcx rbx rbp rsi rdi ✓
    ↴ r8 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 xmm11 xmm12 ✓
    ↴ xmm13 xmm14 xmm15
just_copy2:: ; Preds .B1.4 .B1.6
; R8 = cnt
; RDX = a2
; RCX = a1
    xor      r10d, r10d ✓
    ↴ ;8.2
    xor      r9d, r9d ✓
    ↴ ;
    xor      eax, eax ✓
    ↴ ;
    ; LOE rax rdx rcx rbx rbp rsi ✓
    ↴ rdi r8 r9 r10 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 ✓
    ↴ xmm11 xmm12 xmm13 xmm14 xmm15
.B1.8:: ; Preds .B1.8 just_copy2
    mov      r11d, DWORD PTR [rax+rdx] ✓
    ↴ ;3.6
    inc      r10 ✓
    ↴ ;8.2
    mov      DWORD PTR [r9+rcx], r11d ✓
    ↴ ;3.6
    add      r9, 12 ✓

```

```

    ↴ ;8.2
      add      rax, 28
    ↴ ;8.2
      cmp      r10, r8
    ↴ ;8.2
      jb       .B1.8      ; Prob 82%
    ↴ ;8.2
      jmp      exit       ; Prob 100%
    ↴ ;8.2
                  ; LOE rax rdx rcx rbx rbp rsi ↵
    ↴ rdi r8 r9 r10 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 ↵
    ↴ xmm11 xmm12 xmm13 xmm14 xmm15
just_copy::                                ; Preds .B1.2 .B1.5 .B1.6
; R8 = cnt
; RDX = a2
; RCX = a1
      xor      r10d, r10d
    ↴ ;8.2
      xor      r9d, r9d
    ↴ ;
      xor      eax, eax
    ↴ ;
                  ; LOE rax rdx rcx rbx rbp rsi ↵
    ↴ rdi r8 r9 r10 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 ↵
    ↴ xmm11 xmm12 xmm13 xmm14 xmm15
.B1.11::                                     ; Preds .B1.11 just_copy
      mov      r11d, DWORD PTR [rax+rdx]
    ↴ ;3.6
      inc      r10
    ↴ ;8.2
      mov      DWORD PTR [r9+rcx], r11d
    ↴ ;3.6
      add      r9, 12
    ↴ ;8.2
      add      rax, 28
    ↴ ;8.2
      cmp      r10, r8
    ↴ ;8.2
      jb       .B1.11     ; Prob 82%
    ↴ ;8.2
                  ; LOE rax rdx rcx rbx rbp rsi ↵
    ↴ rdi r8 r9 r10 r12 r13 r14 r15 xmm6 xmm7 xmm8 xmm9 xmm10 ↵
    ↴ xmm11 xmm12 xmm13 xmm14 xmm15
exit::                                       ; Preds .B1.11 .B1.8 .B1.1
      ret
    ↴ ;10.1

```


Capítulo 40

Duff's device

Duff's device

.?

:

•

•

..:

```
#include <stdint.h>
#include <stdio.h>

void bzero(uint8_t* dst, size_t count)
{
    int i;

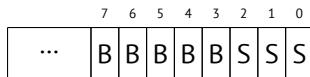
    if (count&(~7))
        //
        for (i=0; i<count>>3; i++)
    {
        *(uint64_t*)dst=0;
        dst=dst+8;
    };

    //
    switch(count & 7)
    {
        case 7: *dst++ = 0;
```

```

        case 6: *dst++ = 0;
        case 5: *dst++ = 0;
        case 4: *dst++ = 0;
        case 3: *dst++ = 0;
        case 2: *dst++ = 0;
        case 1: *dst++ = 0;
        case 0: //break;
    }
}

```



Spanish text placeholder.

:

```

dst$ = 8
count$ = 16
bzero PROC
    test    rdx, -8
    je     SHORT $LN11@bzero
;
    xor    r10d, r10d
    mov    r9, rdx
    shr    r9, 3
    mov    r8d, r10d
    test   r9, r9
    je     SHORT $LN11@bzero
    npad   5
$LL19@bzero:
    inc    r8d
    mov    QWORD PTR [rcx], r10
    add    rcx, 8
    movsxd rax, r8d
    cmp    rax, r9
    jb    SHORT $LL19@bzero
$LN11@bzero:
;
    and   edx, 7
    dec    rdx
    cmp    rdx, 6

```

```

ja    SHORT $LN9@bzero
lea   r8, OFFSET FLAT:__ImageBase
mov   eax, DWORD PTR $LN22@bzero[r8+rdx*4]
add   rax, r8
jmp   rax
$LN8@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN7@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN6@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN5@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN4@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN3@bzero:
    mov   BYTE PTR [rcx], 0
    inc   rcx
$LN2@bzero:
    mov   BYTE PTR [rcx], 0
$LN9@bzero:
    fatret 0
    npad 1
$LN22@bzero:
    DD    $LN2@bzero
    DD    $LN3@bzero
    DD    $LN4@bzero
    DD    $LN5@bzero
    DD    $LN6@bzero
    DD    $LN7@bzero
    DD    $LN8@bzero
bzero ENDP

```

Capítulo 41

División por 9

```
int f(int a)
{
    return a/9;
}
```

41.1 x86

...

Listing 41.1: MSVC

```
_a$ = 8           ; size = 4
_f    PROC
    push   ebp
    mov    ebp, esp
    mov    eax, DWORD PTR _a$[ebp]
    cdq
    mov    ecx, 9
    idiv   ecx
    pop    ebp
    ret    0
_f    ENDP
```

Listing 41.2: Con optimización MSVC

```
_a$ = 8           ; size = 4
_f    PROC
```

```

    mov    ecx, DWORD PTR _a$[esp-4]
    mov    eax, 954437177 ; 38e38e39H
    imul   ecx
    sar    edx, 1
    mov    eax, edx
    shr    eax, 31          ; 0000001fH
    add    eax, edx
    ret    0
_f     ENDP

```

«strength reduction».

Listing 41.3: Sin optimización GCC 4.4.1

```

public f
f      proc near

arg_0 = dword ptr 8

    push    ebp
    mov     ebp, esp
    mov     ecx, [ebp+arg_0]
    mov     edx, 954437177 ; 38E38E39h
    mov     eax, ecx
    imul   edx
    sar    edx, 1
    mov     eax, ecx
    sar    eax, 1Fh
    mov     ecx, edx
    sub    ecx, eax
    mov     eax, ecx
    pop    ebp
    retn
f      endp

```

41.2 ARM

(19 on page 389).

[Ltd94, 3.3 Division by a Constant]. .

```

; takes argument in a1
; returns quotient in a1, remainder in a2
; cycles could be saved if only divide or remainder is required
    SUB    a2, a1, #10           ; keep (x-10) for later

```

```

SUB    a1, a1, a1, lsr #2
ADD    a1, a1, a1, lsr #4
ADD    a1, a1, a1, lsr #8
ADD    a1, a1, a1, lsr #16
MOV    a1, a1, lsr #3
ADD    a3, a1, a1, asl #2
SUBS   a2, a2, a3, asl #1      ; calc (x-10) - (x/10)*10
ADDPL  a1, a1, #1             ; fix-up quotient
ADDMI  a2, a2, #10            ; fix-up remainder
MOV    pc, lr

```

41.2.1 Con optimización Xcode 4.6.3 (LLVM) (Modo ARM)

```

__text:00002C58 39 1E 08 E3 E3 18 43 E3 MOV R1, 0x38E38E39
__text:00002C60 10 F1 50 E7 SMMUL R0, R0, R1
__text:00002C64 C0 10 A0 E1 MOV R1, R0, ASR#1
__text:00002C68 A0 0F 81 E0 ADD R0, R1, R0, LSR#31
    ↴ #31
__text:00002C6C 1E FF 2F E1 BX LR

```

IDA

SMMUL (*Signed Most Significant Word Multiply*)

«MOV R1, R0, ASR#1»

«ADD R0, R1, R0, LSR#31» $R0 = R1 + R0 >> 31$

(MOV, ADD, SUB, RSB)¹ ASR *Arithmetic Shift Right*, LSR *Logical Shift Right*.

41.2.2 Con optimización Xcode 4.6.3 (LLVM) (Modo Thumb-2)

```

MOV      R1, 0x38E38E39
SMMUL.W R0, R0, R1
ASRS    R1, R0, #1
ADD.W   R0, R1, R0, LSR#31
BX      LR

```

, ASRS () .

41.2.3 Sin optimización Xcode 4.6.3 (LLVM) y Keil 6/2013

Sin optimización LLVM .

_aeabi_idivmod.

¹Estas instrucciones también se llaman «data processing instructions»

41.3 MIPS

Listing 41.4: Con optimización GCC 4.4.5 (IDA)

```

f:
    li      $v0, 9
    bnez   $v0, loc_10
    div    $a0, $v0 ; branch delay slot
    break  0x1C00 ; "break 7"

loc_10:
    mflo   $v0
    jr    $ra
    or    $at, $zero ; branch delay slot, NOP

```

«a % 9»,

41.4

:

$$result = \frac{input}{divisor} = \frac{input \cdot \frac{2^n}{divisor}}{2^n} = \frac{input \cdot M}{2^n}$$

M magic.

:

$$M = \frac{2^n}{divisor}$$

:

$$result = \frac{input \cdot M}{2^n}$$

$n < 32$, (en EAX o RAX). $n \geq 32$, (en EDX o RDX).

n .

:

```

int f3_32_signed(int a)
{
    return a/3;
};

unsigned int f3_32_unsigned(unsigned int a)
{
    return a/3;
};

```

magic 0xAAAAAAAB 2^{33} .

magic 0x55555556 2^{32} . EDX.

1 .

Listing 41.5: Con optimización MSVC 2012

```

_f3_32_unsigned PROC
    mov     eax, -1431655765          ; aaaaaaabH
    mul     DWORD PTR _a$[esp-4] ;
; EDX=(input*0xaaaaaaaaab)/2^32
    shr     edx, 1
; EDX=(input*0xaaaaaaaaab)/2^33
    mov     eax, edx
    ret     0
_f3_32_unsigned ENDP

_f3_32_signed PROC
    mov     eax, 1431655766          ; 555555556H
    imul   DWORD PTR _a$[esp-4] ;
;
; 2^32
    mov     eax, edx          ; EAX=EDX=(input*0x55555556) /
    ↴ /2^32
    shr     eax, 31           ; 0000001fH
    add     eax, edx          ;
    ret     0
_f3_32_signed ENDP

```

41.4.1

$$\frac{x}{c} = x \frac{1}{c}$$

[War02, págs. 10-3].

41.5

41.5.1 #1

:

```
    mov      eax, MAGICAL CONSTANT
    imul
    sar      edx, SHIFTING COEFFICIENT ;
    mov      eax, edx
    shr      eax, 31
    add      eax, edx
```

$M, C D.$

:

$$D = \frac{2^{32+C}}{M}$$

:

Listing 41.6: Con optimización MSVC 2012

```
    mov      eax, 2021161081 ; ↵
    ↳ 78787879H
        imul    DWORD PTR _a$[esp-4]
        sar     edx, 3
        mov     eax, edx
        shr     eax, 31 ; ↵
    ↳ 0000001fH
        add     eax, edx
```

:

$$D = \frac{2^{32+3}}{2021161081}$$

Wolfram Mathematica :

²Wikipedia

Listing 41.7: Wolfram Mathematica

```
In[1]:=N[2^(32+3)/2021161081]
Out[1]:=17.
```

17.

 $2^{64} / 2^{32}$:

```
uint64_t f1234(uint64_t a)
{
    return a/1234;
};
```

Listing 41.8: Con optimización MSVC 2012 x64

```
f1234 PROC
    mov     rax, 7653754429286296943          ; 62
    ↳ a37991a23aead6fH
    mul     rcx
    shr     rdx, 9
    mov     rax, rdx
    ret     0
f1234 ENDP
```

Listing 41.9: Wolfram Mathematica

```
In[1]:=N[2^(64+9)/16^^6a37991a23aead6f]
Out[1]:=1234.
```

41.5.2 #2

:

```
        mov     eax, 55555556h ; 1431655766
        imul   ecx
        mov     eax, edx
        shr     eax, 1Fh
```

:

$$D = \frac{2^{32}}{M}$$

:

$$D = \frac{2^{32}}{1431655766}$$

Wolfram Mathematica:

Listing 41.10: Wolfram Mathematica

```
In[1]:=N[2^32/16^^55555556]
Out[1]:=3.
```

3.

41.6 Ejercicio #1

Lo que hace el código?

Listing 41.11: Con optimización MSVC 2010

```
_a$ = 8
_f PROC
    mov     ecx, DWORD PTR _a$[esp-4]
    mov     eax, -968154503 ; c64b2279H
    imul    ecx
    add     edx, ecx
    sar     edx, 9
    mov     eax, edx
    shr     eax, 31          ; 00000001FH
    add     eax, edx
    ret     0
_f ENDP
```

Listing 41.12: Con optimización GCC 4.9 (ARM64)

```
f:
    mov     w1, 8825
    movk   w1, 0xc64b, lsl 16
    smull  x1, w0, w1
    lsr    x1, x1, 32
    add    w1, w0, w1
    asr    w1, w1, 9
    sub    w0, w1, w0, asr 31
    ret
```

Responda [G.1.16 on page 1268](#).

Capítulo 42

(atoi())

42.1

```
#include <stdio.h>

int my_atoi (char *s)
{
    int rt=0;

    while (*s)
    {
        rt=rt*10 + (*s-'0');
        s++;
    };

    return rt;
};

int main()
{
    printf ("%d\n", my_atoi ("1234"));
    printf ("%d\n", my_atoi ("1234567890"));
};
```

42.1.1 Con optimización MSVC 2013 x64

Listing 42.1: Con optimización MSVC 2013 x64

```
s$ = 8
my_atoi PROC
;
    movzx    r8d, BYTE PTR [rcx]
;
;
    xor      eax, eax
;
;
    test     r8b, r8b
    je      SHORT $LN9@my_atoi
$LL2@my_atoi:
    lea      edx, DWORD PTR [rax+rax*4]
; EDX=RAX+RAX*4=rt+rt*4=rt*5
    movsx    eax, r8b
; EAX=
; R8D
    movzx    r8d, BYTE PTR [rcx+1]
; :
    lea      rcx, QWORD PTR [rcx+1]
    lea      eax, DWORD PTR [rax+rdx*2]
; EAX=RAX+RDX*2= + rt*5*2= + rt*10
; 48 (0x30 '0')
    add      eax, -48
    ↴ ffffffffffffffd0H ; ↴
; ?
    test     r8b, r8b
;
    jne     SHORT $LL2@my_atoi
$LN9@my_atoi:
    ret      0
my_atoi ENDP
```

42.1.2 Con optimización GCC 4.9.1 x64

Con optimización GCC 4.9.1 .

Listing 42.2: Con optimización GCC 4.9.1 x64

```
my_atoi:
; EDX
    movsx    edx, BYTE PTR [rdi]
; EAX
    xor      eax, eax
```

```

;
    test    dl, dl
    je     .L4
.L3:
    lea     eax, [rax+rax*4]
; EAX=RAX*5=rt*5
;
    add    rdi, 1
    lea     eax, [rdx-48+rax*2]
; EAX= - 48 + RAX*2 = - '0' + rt*10
;
    movsx   edx, BYTE PTR [rdi]
;
    test    dl, dl
    jne     .L3
    rep ret
.L4:
    rep ret

```

42.1.3 Con optimización Keil 6/2013 (Modo ARM)

Listing 42.3: Con optimización Keil 6/2013 (Modo ARM)

```

my_atoi PROC
; R1
    MOV     r1,r0
; R0
    MOV     r0,#0
    B      |L0.28|
|L0.12|
    ADD     r0,r0,r0,LSL #2
; R0=R0+R0<<2=rt*5
    ADD     r0,r2,r0,LSL #1
; R0= + rt*5<<1 = + rt*10
; '0' rt:
    SUB     r0,r0,#0x30
;
    ADD     r1,r1,#1
|L0.28|
; R2
    LDRB   r2,[r1,#0]
;
    CMP     r2,#0
    BNE     |L0.12|
;

```

BX	lr
ENDP	

42.1.4 Con optimización Keil 6/2013 (Modo Thumb)

Listing 42.4: Con optimización Keil 6/2013 (Modo Thumb)

```

my_atoi PROC
; R1
    MOVS    r1,r0
; R0
    MOVS    r0,#0
    B      |L0.16|
|L0.6|
    MOVS    r3,#0xa
; R3=10
    MULS    r0,r3,r0
; R0=R3*R0=rt*10
; :
    ADDS    r1,r1,#1
; :
    SUBS    r0,r0,#0x30
    ADDS    r0,r2,r0
; rt=R2+R0= + (rt*10 - '0')
|L0.16|
; R2
    LDRB    r2,[r1,#0]
; ?
    CMP     r2,#0
;
    BNE     |L0.6|
;
    BX      lr
ENDP

```

42.1.5 Con optimización GCC 4.9.1 ARM64

Listing 42.5: Con optimización GCC 4.9.1 ARM64

```

my_atoi:
; W1
    ldrb    w1, [x0]
    mov     x2, x0
; X2=

```

```

; ?
;
; W1 .
;
;      cbz      w1,  .L4
; W0
; :
;      mov      w0,  0
.L3:
; 48  '0'  W3:
;      sub      w3,  w1,  #48
; X2+1  W1 :
;      ldrb    w1,  [x2,1]!
;      add     w0,  w0,  w0,  lsl 2
; W0=W0+W0<<2=W0+W0*4=rt*5
;      add     w0,  w3,  w0,  lsl 1
; W0= + W0<<1 = + rt*5*2 = + rt*10
;      cbnz    w1,  .L3
;
;      ret
.L4:
;      mov      w0,  w1
;      ret

```

42.2

```

#include <stdio.h>

int my_atoi (char *s)
{
    int negative=0;
    int rt=0;

    if (*s=='-')
    {
        negative=1;
        s++;
    };

    while (*s)
    {
        if (*s<'0' || *s>'9')
        {
            printf ("Error! Unexpected char: '%c'\n",
                   *s);

```

```

        exit(0);
    };
    rt=rt*10 + (*s-'0');
    s++;
};

if (negative)
    return -rt;
return rt;
};

int main()
{
    printf ("%d\n", my_atoi ("1234"));
    printf ("%d\n", my_atoi ("1234567890"));
    printf ("%d\n", my_atoi ("-1234"));
    printf ("%d\n", my_atoi ("-1234567890"));
    printf ("%d\n", my_atoi ("-a1234567890")); // error
};

```

42.2.1 Con optimización GCC 4.9.1 x64

Listing 42.6: Con optimización GCC 4.9.1 x64

```

.LC0:
.string "Error! Unexpected char: '%c'\n"

my_atoi:
    sub    rsp, 8
    movsx  edx, BYTE PTR [rdi]
;
    cmp    dl, 45 ; '-'
    je     .L22
    xor    esi, esi
    test   dl, dl
    je     .L20
.L10:
; ESI=0
    lea    eax, [rdx-48]
;
;
    cmp    al, 9
    ja    .L4
    xor    eax, eax
    jmp   .L6
.L7:

```

```

        lea      ecx, [rdx-48]
        cmp      cl, 9
        ja       .L4

.L6:
        lea      eax, [rax+rax*4]
        add      rdi, 1
        lea      eax, [rdx-48+rax*2]
        movsx   edx, BYTE PTR [rdi]
        test    dl, dl
        jne     .L7
;

;
; .
        test    esi, esi
        je     .L18
        neg    eax

.L18:
        add    rsp, 8
        ret

.L22:
        movsx   edx, BYTE PTR [rdi+1]
        lea      rax, [rdi+1]
        test    dl, dl
        je     .L20
        mov    rdi, rax
        mov    esi, 1
        jmp     .L10

.L20:
        xor    eax, eax
        jmp     .L18

.L4:
;

        mov    edi, 1
        mov    esi, OFFSET FLAT:.LC0 ; "Error! Unexpected char\x
        ↵ : '%c\n"
        xor    eax, eax
        call   __printf_chk
        xor    edi, edi
        call   exit

```

```

if (*s<'0' || *s>'9')
    ...

```

```

    ,

```

$46 - 48 = -2$. $0xFFFFFFFFE$ (4294967294)

: [48.1.2 on page 684](#).

Con optimización MSVC 2013 x64 .

42.2.2 Con optimización Keil 6/2013 (Modo ARM)

Listing 42.7: Con optimización Keil 6/2013 (Modo ARM)

```

1 my_atoi PROC
2     PUSH    {r4-r6,lr}
3     MOV     r4,r0
4     LDRB   r0,[r0,#0]
5     MOV     r6,#0
6     MOV     r5,r6
7     CMP     r0,#0x2d '-'
8 ; R6
9     MOVEQ  r6,#1
10    ADDEQ  r4,r4,#1
11    B      |L0.80|
12
13 |L0.36|
14    SUB    r0,r1,#0x30
15    CMP    r0,#0xa
16    BCC   |L0.64|
17    ADR    r0,|L0.220|
18    BL     __2printf
19    MOV    r0,#0
20    BL     exit
21
22 |L0.64|
23    LDRB  r0,[r4],#1
24    ADD   r1,r5,r5,LSL #2
25    ADD   r0,r0,r1,LSL #1
26    SUB   r5,r0,#0x30
27
28 |L0.80|
29    LDRB  r1,[r4,#0]
30    CMP   r1,#0
31    BNE   |L0.36|
32    CMP   r6,#0
33 ;
34    RSBNE r0,r5,#0
35    MOVEQ r0,r5
36    POP   {r4-r6,pc}
37    ENDP
38
39 |L0.220|
40    DCB    "Error! Unexpected char: '%c'\n",0

```

: $0 - x = -x$.

GCC 4.9 para ARM64 ARM64.

42.3 Ejercicio

Capítulo 43

Listing 43.1:

```
#include <stdio.h>

int celsius_to_fahrenheit (int celsius)
{
    return celsius * 9 / 5 + 32;
}

int main(int argc, char *argv[])
{
    int celsius=atol(argv[1]);
    printf ("%d\n", celsius_to_fahrenheit (celsius));
}
```

...

Listing 43.2: Con optimización GCC 4.8.1

```
_main:
    push    ebp
    mov     ebp, esp
    and     esp, -16
    sub     esp, 16
    call    __main
    mov     eax, DWORD PTR [ebp+12]
    mov     eax, DWORD PTR [eax+4]
    mov     DWORD PTR [esp], eax
    call    _atol
    mov     edx, 1717986919
    mov     DWORD PTR [esp], OFFSET FLAT:LC2 ; "%d\12\0"
    lea     ecx, [eax+eax*8]
    mov     eax, ecx
    imul   edx
```

```

sar    ecx, 31
sar    edx
sub    edx, ecx
add    edx, 32
mov    DWORD PTR [esp+4], edx
call   _printf
leave
ret

```

((41 on page 631).)

43.1

strcpy(), strcmp(), strlen(), memset(), memcmp(), memcpy(), Spanish text placeholder.

43.1.1 strcmp()

Listing 43.3:

```

bool is_bool (char *s)
{
    if (strcmp (s, "true") == 0)
        return true;
    if (strcmp (s, "false") == 0)
        return false;

    assert(0);
}

```

Listing 43.4: Con optimización GCC 4.8.1

```

.LC0:
    .string "true"
.LC1:
    .string "false"
is_bool:
.LFB0:
    push    edi
    mov     ecx, 5
    push    esi
    mov     edi, OFFSET FLAT:.LC0
    sub     esp, 20
    mov     esi, DWORD PTR [esp+32]
    repz   cmpsb
    je     .L3
    mov     esi, DWORD PTR [esp+32]

```

```

    mov    ecx, 6
    mov    edi, OFFSET FLAT:.LC1
    repz cmpsb
    seta   cl
    setb   dl
    xor    eax, eax
    cmp    cl, dl
    jne    .L8
    add    esp, 20
    pop    esi
    pop    edi
    ret

.L8:
    mov    DWORD PTR [esp], 0
    call   assert
    add    esp, 20
    pop    esi
    pop    edi
    ret

.L3:
    add    esp, 20
    mov    eax, 1
    pop    esi
    pop    edi
    ret

```

Listing 43.5: Con optimización MSVC 2010

```

$SG3454 DB      'true', 00H
$SG3456 DB      'false', 00H

_s$ = 8          ; size = 4
?is_bool@@YA_NPAD@Z PROC ; is_bool
    push   esi
    mov    esi, DWORD PTR _s$[esp]
    mov    ecx, OFFSET $SG3454 ; 'true'
    mov    eax, esi
    npad  4 ;
$LL6@is_bool:
    mov    dl, BYTE PTR [eax]
    cmp    dl, BYTE PTR [ecx]
    jne    SHORT $LN7@is_bool
    test   dl, dl
    je     SHORT $LN8@is_bool
    mov    dl, BYTE PTR [eax+1]
    cmp    dl, BYTE PTR [ecx+1]
    jne    SHORT $LN7@is_bool

```

```

        add    eax, 2
        add    ecx, 2
        test   dl, dl
        jne    SHORT $LL6@is_bool
$LN8@is_bool:
        xor    eax, eax
        jmp    SHORT $LN9@is_bool
$LN7@is_bool:
        sbb    eax, eax
        sbb    eax, -1
$LN9@is_bool:
        test   eax, eax
        jne    SHORT $LN2@is_bool

        mov    al, 1
        pop    esi

        ret    0
$LN2@is_bool:

        mov    ecx, OFFSET $SG3456 ; 'false'
        mov    eax, esi
$LL10@is_bool:
        mov    dl, BYTE PTR [eax]
        cmp    dl, BYTE PTR [ecx]
        jne    SHORT $LN11@is_bool
        test   dl, dl
        je     SHORT $LN12@is_bool
        mov    dl, BYTE PTR [eax+1]
        cmp    dl, BYTE PTR [ecx+1]
        jne    SHORT $LN11@is_bool
        add    eax, 2
        add    ecx, 2
        test   dl, dl
        jne    SHORT $LL10@is_bool
$LN12@is_bool:
        xor    eax, eax
        jmp    SHORT $LN13@is_bool
$LN11@is_bool:
        sbb    eax, eax
        sbb    eax, -1
$LN13@is_bool:
        test   eax, eax
        jne    SHORT $LN1@is_bool

        xor    al, al
        pop    esi

```

```

        ret      0
$LN1@is_bool:

        push    11
        push    OFFSET $SG3458
        push    OFFSET $SG3459
        call    DWORD PTR __imp__wassert
        add     esp, 12
        pop     esi

        ret      0
?is_bool@@YA_NPAD@Z ENDP ; is_bool

```

43.1.2 strlen()

Listing 43.6:

```

int strlen_test(char *s1)
{
    return strlen(s1);
}

```

Listing 43.7: Con optimización MSVC 2010

```

_s1$ = 8 ; size = 4
_strlen_test PROC
    mov     eax, DWORD PTR _s1$[esp-4]
    lea     edx, DWORD PTR [eax+1]
$LL3@strlen_tes:
    mov     cl, BYTE PTR [eax]
    inc     eax
    test    cl, cl
    jne    SHORT $LL3@strlen_tes
    sub     eax, edx
    ret     0
_strlen_test ENDP

```

43.1.3 strcpy()

Listing 43.8:

```

void strcpy_test(char *s1, char *outbuf)
{
    strcpy(outbuf, s1);
}

```

Listing 43.9: Con optimización MSVC 2010

```

_s1$ = 8          ; size = 4
_outbuf$ = 12    ; size = 4
_strcmpy_test PROC
    mov     eax, DWORD PTR _s1$[esp-4]
    mov     edx, DWORD PTR _outbuf$[esp-4]
    sub     edx, eax
    npad    6 ;
$LL3@strcmpy_tes:
    mov     cl, BYTE PTR [eax]
    mov     BYTE PTR [edx+eax], cl
    inc     eax
    test    cl, cl
    jne     SHORT $LL3@strcmpy_tes
    ret     0
_strcmpy_test ENDP

```

43.1.4 memset()

Ejemplo#1

Listing 43.10: 32

```

#include <stdio.h>

void f(char *out)
{
    memset(out, 0, 32);
}

```

Listing 43.11: Con optimización GCC 4.9.1 x64

```

f:
    mov     QWORD PTR [rdi], 0
    mov     QWORD PTR [rdi+8], 0
    mov     QWORD PTR [rdi+16], 0
    mov     QWORD PTR [rdi+24], 0
    ret

```

[: 14.1.4 on page 228.](#)

Ejemplo#2

Listing 43.12: 67

```
#include <stdio.h>

void f(char *out)
{
    memset(out, 0, 67);
}
```

Listing 43.13: Con optimización MSVC 2012 x64

```
out$ = 8
f      PROC
        xor    eax, eax
        mov    QWORD PTR [rcx], rax
        mov    QWORD PTR [rcx+8], rax
        mov    QWORD PTR [rcx+16], rax
        mov    QWORD PTR [rcx+24], rax
        mov    QWORD PTR [rcx+32], rax
        mov    QWORD PTR [rcx+40], rax
        mov    QWORD PTR [rcx+48], rax
        mov    QWORD PTR [rcx+56], rax
        mov    WORD PTR [rcx+64], ax
        mov    BYTE PTR [rcx+66], al
        ret    0
f      ENDP
```

Listing 43.14: Con optimización GCC 4.9.1 x64

```
f:
        mov    QWORD PTR [rdi], 0
        mov    QWORD PTR [rdi+59], 0
        mov    rcx, rdi
        lea    rdi, [rdi+8]
        xor    eax, eax
        and    rdi, -8
        sub    rcx, rdi
        add    ecx, 67
        shr    ecx, 3
        rep    stosq
        ret
```

43.1.5 memcpy()

Listing 43.15:

```
void memcpy_7(char *inbuf, char *outbuf)
```

```
{
    memcpy(outbuf+10, inbuf, 7);
};
```

Listing 43.16: Con optimización MSVC 2010

```
_inbuf$ = 8      ; size = 4
_outbuf$ = 12    ; size = 4
_memcpy_7 PROC
    mov     ecx, DWORD PTR _inbuf$[esp-4]
    mov     edx, DWORD PTR [ecx]
    mov     eax, DWORD PTR _outbuf$[esp-4]
    mov     DWORD PTR [eax+10], edx
    mov     dx, WORD PTR [ecx+4]
    mov     WORD PTR [eax+14], dx
    mov     cl, BYTE PTR [ecx+6]
    mov     BYTE PTR [eax+16], cl
    ret     0
_memcpy_7 ENDP
```

Listing 43.17: Con optimización GCC 4.8.1

```
memcpy_7:
    push    ebx
    mov     eax, DWORD PTR [esp+8]
    mov     ecx, DWORD PTR [esp+12]
    mov     ebx, DWORD PTR [eax]
    lea     edx, [ecx+10]
    mov     DWORD PTR [ecx+10], ebx
    movzx  ecx, WORD PTR [eax+4]
    mov     WORD PTR [edx+4], cx
    movzx  eax, BYTE PTR [eax+6]
    mov     BYTE PTR [edx+6], al
    pop    ebx
    ret
```

: 21.4.1 on page 469.

Listing 43.18:

```
void memcpy_128(char *inbuf, char *outbuf)
{
    memcpy(outbuf+10, inbuf, 128);
};
```

```
void memcpy_123(char *inbuf, char *outbuf)
{
    memcpy(outbuf+10, inbuf, 123);
}
```

Listing 43.19: Con optimización MSVC 2010

```
_inbuf$ = 8           ; size = 4
_outbuf$ = 12          ; size = 4
_memcpy_128 PROC
    push    esi
    mov     esi, DWORD PTR _inbuf$[esp]
    push    edi
    mov     edi, DWORD PTR _outbuf$[esp+4]
    add     edi, 10
    mov     ecx, 32
    rep    movsd
    pop     edi
    pop     esi
    ret     0
_memcpy_128 ENDP
```

Listing 43.20: Con optimización MSVC 2010

```
_inbuf$ = 8           ; size = 4
_outbuf$ = 12          ; size = 4
_memcpy_123 PROC
    push    esi
    mov     esi, DWORD PTR _inbuf$[esp]
    push    edi
    mov     edi, DWORD PTR _outbuf$[esp+4]
    add     edi, 10
    mov     ecx, 30
    rep    movsd
    movsw
    movsb
    pop     edi
    pop     esi
    ret     0
_memcpy_123 ENDP
```

Listing 43.21: Con optimización GCC 4.8.1

```
memcpy_123:
.LFB3:
    push    edi
    mov     eax, 123
```

```

push    esi
mov     edx, DWORD PTR [esp+16]
mov     esi, DWORD PTR [esp+12]
lea     edi, [edx+10]
test    edi, 1
jne     .L24
test    edi, 2
jne     .L25
.L7:
    mov     ecx, eax
    xor     edx, edx
    shr     ecx, 2
    test    al, 2
    rep movsd
    je      .L8
    movzx  edx, WORD PTR [esi]
    mov    WORD PTR [edi], dx
    mov    edx, 2
.L8:
    test    al, 1
    je      .L5
    movzx  eax, BYTE PTR [esi+edx]
    mov    BYTE PTR [edi+edx], al
.L5:
    pop    esi
    pop    edi
    ret
.L24:
    movzx  eax, BYTE PTR [esi]
    lea     edi, [edx+11]
    add    esi, 1
    test    edi, 2
    mov    BYTE PTR [edx+10], al
    mov    eax, 122
    je      .L7
.L25:
    movzx  edx, WORD PTR [esi]
    add    edi, 2
    add    esi, 2
    sub    eax, 2
    mov    WORD PTR [edi-2], dx
    jmp    .L7
.LFE3:

```

: 25.2 on page 546.

43.1.6 memcmp()

Listing 43.22:

```
void memcpy_1235(char *inbuf, char *outbuf)
{
    memcpy(outbuf+10, inbuf, 1235);
}
```

Listing 43.23: Con optimización MSVC 2010

```
_buf1$ = 8      ; size = 4
_buf2$ = 12      ; size = 4
_memcpy_1235 PROC
    mov     edx, DWORD PTR _buf2$[esp-4]
    mov     ecx, DWORD PTR _buf1$[esp-4]
    push    esi
    push    edi
    mov     esi, 1235
    add     edx, 10
$LL4@memcpy_123:
    mov     eax, DWORD PTR [edx]
    cmp     eax, DWORD PTR [ecx]
    jne    SHORT $LN10@memcpy_123
    sub     esi, 4
    add     ecx, 4
    add     edx, 4
    cmp     esi, 4
    jae    SHORT $LL4@memcpy_123
$LN10@memcpy_123:
    movzx   edi, BYTE PTR [ecx]
    movzx   eax, BYTE PTR [edx]
    sub     eax, edi
    jne    SHORT $LN7@memcpy_123
    movzx   eax, BYTE PTR [edx+1]
    movzx   edi, BYTE PTR [ecx+1]
    sub     eax, edi
    jne    SHORT $LN7@memcpy_123
    movzx   eax, BYTE PTR [edx+2]
    movzx   edi, BYTE PTR [ecx+2]
    sub     eax, edi
    jne    SHORT $LN7@memcpy_123
    cmp     esi, 3
    jbe    SHORT $LN6@memcpy_123
    movzx   eax, BYTE PTR [edx+3]
    movzx   ecx, BYTE PTR [ecx+3]
    sub     eax, ecx
$LN7@memcpy_123:
```

```
sar    eax, 31
pop    edi
or     eax, 1
pop    esi
ret    0
$LN6@memcmp_123:
    pop   edi
    xor   eax, eax
    pop   esi
    ret   0
 memcmp_1235 ENDP
```

43.1.7

También hay un pequeño script para [IDA](#) que busca y contrae estos fragmentos de código inline tan frecuentes.

[GitHub](#).

Capítulo 44

C99 restrict

```
void f1 (int* x, int* y, int* sum, int* product, int* ↴
     ↴ sum_product, int* update_me, size_t s)
{
    for (int i=0; i<s; i++)
    {
        sum[i]=x[i]+y[i];
        product[i]=x[i]*y[i];
        update_me[i]=i*123; // some dummy value
        sum_product[i]=sum[i]+product[i];
    };
}
```

: update_me sum, product, sum_product

- sum[i]
- product[i]
- update_me[i]
- sum_product[i] sum[i] y product[i]

sum[i] y product[i] «pointer aliasing»,

restrict [ISO07, págs. 6.7.3/1]

restrict.

:

```
void f2 (int* restrict x, int* restrict y, int* restrict sum, ↴
     ↴ int* restrict product, int* restrict sum_product,
            int* restrict update_me, size_t s)
```

```
{
    for (int i=0; i<s; i++)
    {
        sum[i]=x[i]+y[i];
        product[i]=x[i]*y[i];
        update_me[i]=i*123; // some dummy value
        sum_product[i]=sum[i]+product[i];
    };
}
```

:

Listing 44.1: GCC x64: f1()

```
f1:
    push    r15 r14 r13 r12 rbp rdi rsi rbx
    mov     r13, QWORD PTR 120[rsp]
    mov     rbp, QWORD PTR 104[rsp]
    mov     r12, QWORD PTR 112[rsp]
    test   r13, r13
    je      .L1
    add    r13, 1
    xor    ebx, ebx
    mov    edi, 1
    xor    r11d, r11d
    jmp   .L4

.L6:
    mov    r11, rdi
    mov    rdi, rax

.L4:
    lea    rax, 0[0+r11*4]
    lea    r10, [rcx+rax]
    lea    r14, [rdx+rax]
    lea    rsi, [r8+rax]
    add    rax, r9
    mov    r15d, DWORD PTR [r10]
    add    r15d, DWORD PTR [r14]
    mov    DWORD PTR [rsi], r15d      ; sum[]
    mov    r10d, DWORD PTR [r10]
    imul   r10d, DWORD PTR [r14]
    mov    DWORD PTR [rax], r10d      ; product[]
    mov    DWORD PTR [r12+r11*4], ebx ; update_me[]
    add    ebx, 123
    mov    r10d, DWORD PTR [rsi]      ; sum[i]
    add    r10d, DWORD PTR [rax]      ; product[i]
    lea    rax, 1[rdi]
    cmp    rax, r13
```

```

        mov     DWORD PTR 0[rbp+r11*4], r10d ; sum_product[]
        jne     .L6
.L1:
        pop    rbx rsi rdi rbp r12 r13 r14 r15
        ret

```

Listing 44.2: GCC x64: f2()

```

f2:
        push   r13 r12 rbp rdi rsi rbx
        mov    r13, QWORD PTR 104[rsp]
        mov    rbp, QWORD PTR 88[rsp]
        mov    r12, QWORD PTR 96[rsp]
        test   r13, r13
        je     .L7
        add    r13, 1
        xor    r10d, r10d
        mov    edi, 1
        xor    eax, eax
        jmp   .L10
.L11:
        mov    rax, rdi
        mov    rdi, r11
.L10:
        mov    esi, DWORD PTR [rcx+rax*4]
        mov    r11d, DWORD PTR [rdx+rax*4]
        mov    DWORD PTR [r12+rax*4], r10d ; update_me[]
        add    r10d, 123
        lea    ebx, [rsi+r11]
        imul  r11d, esi
        mov    DWORD PTR [r8+rax*4], ebx ; sum[]
        mov    DWORD PTR [r9+rax*4], r11d ; product[]
        add    r11d, ebx
        mov    DWORD PTR 0[rbp+rax*4], r11d ; sum_product[]
        lea    r11, 1[rdi]
        cmp    r11, r13
        jne   .L11
.L7:
        pop    rbx rsi rdi rbp r12 r13
        ret

```

: en f1(), sum[i] y product[i] , f2() , sum[i] y product[i] , .

FORTRAN. , restrict en, FORTRAN .

? [Loh10].

Capítulo 45

```
int my_abs (int i)
{
    if (i<0)
        return -i;
    else
        return i;
};
```

45.1 Con optimización GCC 4.9.1 x64

Listing 45.1: Con optimización GCC 4.9 x64

```
my_abs:
    mov     edx, edi
    mov     eax, edi
    sar     edx, 31
;
;
;
;
    xor     eax, edx
    sub     eax, edx
    ret
```

[War02], págs. 2-4].

45.2 Con optimización GCC 4.9 ARM64

Listing 45.2: Con optimización GCC 4.9 ARM64

```
my_abs:  
;  
    sxtw    x0, w0  
    eor     x1, x0, x0, asr 63  
; X1=X0^(X0>>63)  
    sub     x0, x1, x0, asr 63  
; X0=X1-(X0>>63)=X0^(X0>>63)-(X0>>63)  
    ret
```

Capítulo 46

46.1

```
#include <stdio.h>
#include <stdarg.h>

int arith_mean(int v, ...)
{
    va_list args;
    int sum=v, count=1, i;
    va_start(args, v);

    while(1)
    {
        i=va_arg(args, int);
        if (i==-1) //
            break;
        sum=sum+i;
        count++;
    }

    va_end(args);
    return sum/count;
};

int main()
{
    printf ("%d\n", arith_mean (1, 2, 7, 10, 15, -1 /* */) );
}
```

?

46.1.1

Listing 46.1: Con optimización MSVC 6.0

```

_v$ = 8
_arith_mean PROC NEAR
    mov     eax, DWORD PTR _v$[esp-4] ; sum
    push    esi
    mov     esi, 1                      ; count=1
    lea     edx, DWORD PTR _v$[esp]    ;
$L838:
    mov     ecx, DWORD PTR [edx+4]    ;
    add     edx, 4                   ;
    cmp     ecx, -1                  ; -1?
    je      SHORT $L856            ;
    add     eax, ecx                ; sum = sum +
    inc     esi                     ; count++
    jmp     SHORT $L838            ;
$L856:
;

    cdq
    idiv    esi
    pop     esi
    ret     0
_arith_mean ENDP

$SG851  DB      '%d', 0aH, 00H

_main   PROC NEAR
    push    -1
    push    15
    push    10
    push    7
    push    2
    push    1
    call    _arith_mean
    push    eax
    push    OFFSET FLAT:$SG851 ; '%d'
    call    _printf
    add     esp, 32
    ret     0
_main   ENDP

```

46.1.2

:

Listing 46.2: Con optimización MSVC 2012 x64

```
$SG3013 DB      '%d', 0aH, 00H

v$ = 8
arith_mean PROC
    mov     DWORD PTR [rsp+8], ecx      ;
    mov     QWORD PTR [rsp+16], rdx      ;
    mov     QWORD PTR [rsp+24], r8       ;
    mov     eax, ecx                   ; sum =
    lea     rcx, QWORD PTR v$[rsp+8]   ;
    mov     QWORD PTR [rsp+32], r9       ;
    mov     edx, DWORD PTR [rcx]       ;
    mov     r8d, 1                     ; count=1
    cmp     edx, -1                  ; -1?
    je     SHORT $LN8@arith_mean     ;
$LL3@arith_mean:
    add     eax, edx                 ; sum = sum +
    mov     edx, DWORD PTR [rcx+8]   ;
    lea     rcx, QWORD PTR [rcx+8]   ;
    inc     r8d                      ; count++
    cmp     edx, -1                  ; -1?
    jne     SHORT $LL3@arith_mean     ;
$LN8@arith_mean:
;
    cdq
    idiv    r8d
    ret     0
arith_mean ENDP

main    PROC
    sub     rsp, 56
    mov     edx, 2
    mov     DWORD PTR [rsp+40], -1
    mov     DWORD PTR [rsp+32], 15
    lea     r9d, QWORD PTR [rdx+8]
    lea     r8d, QWORD PTR [rdx+5]
    lea     ecx, QWORD PTR [rdx-1]
    call    arith_mean
    lea     rcx, OFFSET FLAT:$SG3013
    mov     edx, eax
    call    printf
    xor     eax, eax
    add     rsp, 56
```

ret 0
main ENDP

Listing 46.3: Con optimización GCC 4.9.1 x64

```

arith_mean:
    lea    rax, [rsp+8]
    ;
    mov    QWORD PTR [rsp-40], rsi
    mov    QWORD PTR [rsp-32], rdx
    mov    QWORD PTR [rsp-16], r8
    mov    QWORD PTR [rsp-24], rcx
    mov    esi, 8
    mov    QWORD PTR [rsp-64], rax
    lea    rax, [rsp-48]
    mov    QWORD PTR [rsp-8], r9
    mov    DWORD PTR [rsp-72], 8
    lea    rdx, [rsp+8]
    mov    r8d, 1
    mov    QWORD PTR [rsp-56], rax
    jmp   .L5

.L7:
    ;
    lea    rax, [rsp-48]
    mov    ecx, esi
    add    esi, 8
    add    rcx, rax
    mov    ecx, DWORD PTR [rcx]
    cmp    ecx, -1
    je     .L4

.L8:
    add    edi, ecx
    add    r8d, 1

.L5:
    ;
    ;
    cmp    esi, 47
    jbe   .L7           ;
    ;
    mov    rcx, rdx
    add    rdx, 8
    mov    ecx, DWORD PTR [rcx]
    cmp    ecx, -1
    jne   .L8

.L4:
    mov    eax, edi
    cdq

```

```

        idiv    r8d
        ret

.LC1:
        .string "%d\n"
main:
        sub     rsp, 8
        mov     edx, 7
        mov     esi, 2
        mov     edi, 1
        mov     r9d, -1
        mov     r8d, 15
        mov     ecx, 10
        xor     eax, eax
        call    arith_mean
        mov     esi, OFFSET FLAT:.LC1
        mov     edx, eax
        mov     edi, 1
        xor     eax, eax
        add     rsp, 8
        jmp    __printf_chk

```

: 64.8 on page 876.

46.2

?

```

#include <stdlib.h>
#include <stdarg.h>

void die (const char * fmt, ...)
{
    va_list va;
    va_start (va, fmt);

    vprintf (fmt, va);
    exit(0);
}

```

:

Listing 46.4: Con optimización MSVC 2010

```

_fmt$ = 8

```

```

_die PROC
;
    mov     ecx, DWORD PTR _fmt$[esp-4]
;
    lea     eax, DWORD PTR _fmt$[esp]
    push    eax
    push    ecx
    call    _vprintf
    add    esp, 8
    push    0
    call    _exit
$LN3@die:
    int    3
_die ENDP

```

Listing 46.5: Con optimización MSVC 2012 x64

```

fmt$ = 48
die PROC
; Shadow Space
    mov     QWORD PTR [rsp+8], rcx
    mov     QWORD PTR [rsp+16], rdx
    mov     QWORD PTR [rsp+24], r8
    mov     QWORD PTR [rsp+32], r9
    sub    rsp, 40
    lea     rdx, QWORD PTR fmt$[rsp+8] ;
; RCX die()
; vprintf() RCX
    call    vprintf
    xor    ecx, ecx
    call    exit
    int    3
die ENDP

```

Capítulo 47

```
#include <stdio.h>
#include <string.h>

char* str_trim (char *s)
{
    char c;
    size_t str_len;

    // \r \n
    //
    // ()
    for (str_len=strlen(s); str_len>0 && (c=s[str_len-1]); ↴
        str_len--)
    {
        if (c=='\r' || c=='\n')
            s[str_len-1]=0;
        else
            break;
    };
    return s;
};

int main()
{
    //
    //
    //
    //

    printf ("[%s]\n", str_trim (strdup(""))));
```

```

printf ("%[s]\n", str_trim (strdup("\n")));
printf ("%[s]\n", str_trim (strdup("\r")));
printf ("%[s]\n", str_trim (strdup("\n\r")));
printf ("%[s]\n", str_trim (strdup("\r\n")));
printf ("%[s]\n", str_trim (strdup("test1\r\n")));
printf ("%[s]\n", str_trim (strdup("test2\n\r")));
printf ("%[s]\n", str_trim (strdup("test3\n\r\n\r")));
printf ("%[s]\n", str_trim (strdup("test4\n")));
printf ("%[s]\n", str_trim (strdup("test5\r")));
printf ("%[s]\n", str_trim (strdup("test6\r\r\r")));
};

}

```

47.1 x64: Con optimización MSVC 2013

Listing 47.1: Con optimización MSVC 2013 x64

```

s$ = 8
str_trim PROC

; RCX

; :
; RAX 0xFFFFFFFFFFFFFF (-1)
    or     rax, -1
$LL14@str_trim:
    inc    rax
    cmp    BYTE PTR [rcx+rax], 0
    jne    SHORT $LL14@str_trim
;
    test   eax, eax
$LN18@str_trim:
    je     SHORT $LN15@str_trim
; RAX
;
;
    lea    edx, DWORD PTR [rax-1]
;
    mov    eax, edx
; s[str_len-1]
    movzx  eax, BYTE PTR [rdx+rcx]
; R8
    lea    r8, QWORD PTR [rdx+rcx]
    cmp    al, 13 ; is it '\r'?
    je     SHORT $LN2@str_trim
    cmp    al, 10 ; is it '\n'?

```

```

        jne      SHORT $LN15@str_trim
$LN2@str_trim:
;
        mov      BYTE PTR [r8], 0
        mov      eax, edx
;
        test     edx, edx
        jmp      SHORT $LN18@str_trim
$LN15@str_trim:
; "s"
        mov      rax, rcx
        ret      0
str_trim ENDP

```

:43 on page 648.

OR RAX, 0xFFFFFFFFFFFFFF.

..

Listing 47.2: MSVC 2013 x64

```

; RCX =
; RAX =
        or      rax, -1
label:
        inc     rax
        cmp     BYTE PTR [rcx+rax], 0
        jne     SHORT label
; RAX =

```

:

Listing 47.3: strlen()

```

; RCX =
; RAX =
        xor     rax, rax
label:
        cmp     byte ptr [rcx+rax], 0
        jz      exit
        inc     rax
        jmp     label
exit:
; RAX =

```

. «\$LN18@str_trim» JE !

[33.1 on page 589.](#)

47.2 x64: Sin optimización GCC 4.9.1

```

str_trim:
    push    rbp
    mov     rbp,  rsp
    sub    rsp,  32
    mov    QWORD PTR [rbp-24], rdi
;
    mov    rax, QWORD PTR [rbp-24]
    mov    rdi,  rax
    call   strlen
    mov    QWORD PTR [rbp-8], rax ; str_len
;
    jmp    .L2
;
.L5:
    cmp    BYTE PTR [rbp-9], 13 ; c=='\r'?
    je     .L3
    cmp    BYTE PTR [rbp-9], 10 ; c=='\n'?
    jne   .L4
.L3:
    mov    rax, QWORD PTR [rbp-8] ; str_len
    lea    rdx, [rax-1] ; EDX=str_len-1
    mov    rax, QWORD PTR [rbp-24] ; s
    add    rax, rdx ; RAX=s+str_len-1
    mov    BYTE PTR [rax], 0 ; s[str_len-1]=0
;
;
    sub    QWORD PTR [rbp-8], 1 ; str_len--
;
.L2:
;
    cmp    QWORD PTR [rbp-8], 0 ; str_len==0?
    je     .L4
;
    mov    rax, QWORD PTR [rbp-8] ; RAX=str_len
    lea    rdx, [rax-1] ; RDX=str_len-1
    mov    rax, QWORD PTR [rbp-24] ; RAX=s
    add    rax, rdx ; RAX=s+str_len-1
    movzx  eax, BYTE PTR [rax] ; AL=s[str_len-1]
    mov    BYTE PTR [rbp-9], al ; "c"
    cmp    BYTE PTR [rbp-9], 0 ; ?
    jne   .L5
;
```

```
;;
.L4:
;   "s"
    mov      rax, QWORD PTR [rbp-24]
    leave
    ret
```

-
- goto L2
- L5:
-
- L2:
- L4://
- return s

47.3 x64: Con optimización GCC 4.9.1

```
str_trim:
    push    rbx
    mov     rbx, rdi
; RBX "s"
    call    strlen
; str_len==0
    test   rax, rax
    je     .L9
    lea    rdx, [rax-1]
; RDX str_len-1 str_len
;
    lea    rsi, [rbx+rdx]      ; RSI=s+str_len-1
    movzx  ecx, BYTE PTR [rsi] ;
    test   cl, cl
    je     .L9                 ;
    cmp   cl, 10
    je     .L4
    cmp   cl, 13                 ; '\n' '\r'
    jne   .L9
.L4:
;
; MOV RSI, EBX / DEC RSI
```

```

;
    sub    rsi, rax
; RSI = s+str_len-1-str_len = s-1
;
.L12:
    test   rdx, rdx
; s-1+str_len-1+1 = s-1+str_len = s+str_len-1
    mov    BYTE PTR [rsi+1+rdx], 0
; str_len-1==0. .
    je     .L9
    sub    rdx, 1                      ; str_len--
; s+str_len-1
    movzx  ecx, BYTE PTR [rbx+rdx]
    test   cl, cl                     ;
    je     .L9
    cmp    cl, 10                     ; '\n'?
    je     .L12
    cmp    cl, 13                     ; '\r'?
    je     .L12
.L9:
; "s"
    mov    rax, rbx
    pop    rbx
    ret

```

- . ?
- () CR¹/LF²
 -
 -
 -
 - 0
 -
 -

47.4 ARM64: Sin optimización GCC (Linaro) 4.9

:

¹Carriage return (13 o'\r' en C/C++)

²Line feed (10 o'\n' en C/C++)

Listing 47.4: Sin optimización GCC (Linaro) 4.9

```

str_trim:
    stp      x29, x30, [sp, -48]!
    add      x29, sp, 0
    str      x0, [x29,24] ;
    ldr      x0, [x29,24] ; s
    bl       strlen
    str      x0, [x29,40] ; str_len
    b       .L2
;
.L5:
    ldrb    w0, [x29,39]
; W0=c
    cmp     w0, 13          ; '\r'?
    beq     .L3
    ldrb    w0, [x29,39]
; W0=c
    cmp     w0, 10          ; '\n'?
    bne     .L4
;
.L3:
    ldr     x0, [x29,40]
; X0=str_len
    sub     x0, x0, #1
; X0=str_len-1
    ldr     x1, [x29,24]
; X1=s
    add     x0, x1, x0
; X0=s+str_len-1
    strb   wzr, [x0]       ; s+str_len-1
; str_len:
    ldr     x0, [x29,40]
; X0=str_len
    sub     x0, x0, #1
; X0=str_len-1
    str     x0, [x29,40]
;
.L2:
    ldr     x0, [x29,40]
; str_len==0?
    cmp     x0, xzr
;
    beq     .L4
    ldr     x0, [x29,40]
; X0=str_len
    sub     x0, x0, #1
; X0=str_len-1

```

```

        ldr      x1, [x29,24]
; X1=s
        add      x0, x1, x0
; X0=s+str_len-1
; s+str_len-1  W0
        ldrb    w0, [x0]
        strb    w0, [x29,39] ; "c"
        ldrb    w0, [x29,39] ;
; ?
        cmp      w0, wzr
;
        bne      .L5
.L4:
; s
        ldr      x0, [x29,24]
        ldp      x29, x30, [sp], 48
        ret

```

47.5 ARM64: Con optimización GCC (Linaro) 4.9

Listing 47.5: Con optimización GCC (Linaro) 4.9

```

str_trim:
        stp      x29, x30, [sp, -32]!
        add      x29, sp, 0
        str      x19, [sp,16]
        mov      x19, x0
; X19 "s"
        bl       strlen
; X0=str_len
        cbz      x0, .L9          ; str_len==0
        sub      x1, x0, #1
; X1=X0-1=str_len-1
        add      x3, x19, x1
; X3=X19+X1=s+str_len-1
        ldrb    w2, [x19,x1]     ; X19+X1=s+str_len-1
; W2=
        cbz      w2, .L9          ;
        cmp      w2, 10           ; '\n'?
        bne      .L15
.L12:
;
        sub      x2, x1, x0

```

```

; X2=X1-X0=str_len-1-str_len=-1
    add    x2, x3, x2
; X2=X3+X2=s+str_len-1+(-1)=s+str_len-2
    strb   wzr, [x2,1] ; s+str_len-2+1=s+str_len-1
    cbz    x1, .L9      ; str_len-1==0?
    sub    x1, x1, #1   ; str_len--
    ldrb   w2, [x19,x1] ; X19+X1=s+str_len-1
    cmp    w2, 10        ; '\n'?
    cbz    w2, .L9      ;
    beq    .L12         ; '\n'
.L15:
    cmp    w2, 13        ; '\r'?
    beq    .L12         ;
.L9:
; "s"
    mov    x0, x19
    ldr    x19, [sp,16]
    ldp    x29, x30, [sp], 32
    ret

```

47.6 ARM: Con optimización Keil 6/2013 (Modo ARM)

Listing 47.6: Con optimización Keil 6/2013 (Modo ARM)

```

str_trim PROC
    PUSH   {r4,lr}
; R0=s
    MOV    r4,r0
; R4=str_len
    BL     strlen      ; strlen()
; R0=str_len
    MOV    r3,#0
; R3=0
|L0.16|
    CMP    r0,#0       ; str_len==0?
    ADDNE r2,r4,r0    ; ( str_len!=0) R2=R4+R0=s+⤦
    ↴ str_len
    LDRBNE r1,[r2,#-1] ; ( str_len!=0) R1= R2-1=s+⤦
    ↴ str_len-1
    CMPNE r1,#0        ; ( str_len!=0) 0
    BEQ    |L0.56|      ; str_len==0 0
    CMP    r1,#0xd      ; ('\r') '\r'?
    CMPNE r1,#0xa      ; ( '\r' '\n') '\r'?
    SUBEQ r0,r0,#1     ; ( '\r' '\n') R0-- str_len--
    STRBEQ r3,[r2,#-1] ; ( '\r' '\n') R2-1=s+str_len-1

```

```

    BEQ      |L0.16|      ; '\r'  '\n'
|L0.56|
; "s"
    MOV      r0,r4
    POP      {r4,pc}
    ENDP

```

47.7 ARM: Con optimización Keil 6/2013 (Modo Thumb)

?

Listing 47.7: Con optimización Keil 6/2013 (Modo Thumb)

```

str_trim PROC
    PUSH    {r4,lr}
    MOVS    r4,r0
; R4=s
    BL      strlen      ; strlen()
; R0=str_len
    MOVS    r3,#0
; R3 0
    B       |L0.24|
|L0.12|
    CMP     r1,#0xd      ; '\r'?
    BEQ     |L0.20|
    CMP     r1,#0xa      ; '\n'?
    BNE     |L0.38|        ;
|L0.20|
    SUBS   r0,r0,#1      ; R0-- str_len--
    STRB   r3,[r2,#0x1f] ; 0 R2+0x1F=s+str_len-0x20+0x20
    ↳ x1F=s+str_len-1
|L0.24|
    CMP     r0,#0          ; str_len==0?
    BEQ     |L0.38|        ;
    ADDS   r2,r4,r0        ; R2=R4+R0=s+str_len
    SUBS   r2,r2,#0x20      ; R2=R2-0x20=s+str_len-0x20
    LDRB   r1,[r2,#0x1f]    ; R2+0x1F=s+str_len-0x20+0x1F=s
    ↳ +str_len-1 R1
    CMP     r1,#0          ; 0?
    BNE     |L0.12|        ;
|L0.38|
; "s"
    MOVS    r0,r4
    POP      {r4,pc}
    ENDP

```

47.8 MIPS

Listing 47.8: Con optimización GCC 4.4.5 (IDA)

```

str_trim:
; :
saved_GP          = -0x10
saved_S0          = -8
saved_RA          = -4

        lui      $gp, (__gnu_local_gp >> 16)
        addiu   $sp, -0x20
        la      $gp, (__gnu_local_gp & 0xFFFF)
        sw      $ra, 0x20+saved_RA($sp)
        sw      $s0, 0x20+saved_S0($sp)
        sw      $gp, 0x20+saved_GP($sp)
; strlen(). $a0, strlen() :
        lw      $t9, (strlen & 0xFFFF)($gp)
        or      $at, $zero ; load delay slot, NOP
        jalr   $t9
; $a0, $s0:
        move   $s0, $a0    ; branch delay slot
; strlen() ()
; $v0==0 ( 0):
        beqz   $v0, exit
        or      $at, $zero ; branch delay slot, NOP
        addiu  $a1, $v0, -1
; $a1 = $v0-1 = str_len-1
        addu   $a1, $s0, $a1
; $a1 = + $a1 = s+strlen-1
; $a1:
        lb      $a0, 0($a1)
        or      $at, $zero ; load delay slot, NOP
; :
        beqz   $a0, exit
        or      $at, $zero ; branch delay slot, NOP
        addiu  $v1, $v0, -2
; $v1 = str_len-2
        addu   $v1, $s0, $v1
; $v1 = $s0+$v1 = s+str_len-2
        li      $a2, 0xD
; :
        b      loc_6C
        li      $a3, 0xA    ; branch delay slot
loc_5C:
; $a0:
        lb      $a0, 0($v1)

```

```

                    move    $a1, $v1
; $a1=s+str_len-2
; :
                    beqz   $a0, exit
; str_len:
                    addiu  $v1, -1      ; branch delay slot
loc_6C:
; , $a0=, $a2=0xD () $a3=0xA ()
; :
                    beq    $a0, $a2, loc_7C
                    addiu $v0, -1      ; branch delay slot
; :
                    bne    $a0, $a3, exit
                    or     $at, $zero ; branch delay slot, NOP
loc_7C:
;
; :
                    bnez   $v0, loc_5C
; :
                    sb     $zero, 0($a1) ; branch delay slot
; :
exit:
                    lw     $ra, 0x20+saved_RA($sp)
                    move  $v0, $s0
                    lw     $s0, 0x20+saved_S0($sp)
                    jr    $ra
                    addiu $sp, 0x20      ; branch delay slot

```

Capítulo 48

```
char toupper (char c)
{
    if(c>='a' && c<='z')
        return c-'a'+'A';
    else
        return c;
}
```

1.

Characters in the coded character set ascii.															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x C-@	C-a	C-b	C-c	C-d	C-e	C-f	C-g	C-h	TAB	C-j	C-k	C-l	RET	C-n	C-o
1x C-p	C-q	C-r	C-s	C-t	C-u	C-v	C-w	C-x	C-y	C-z	ESC	C-\	C-]	C-^	C-_
2x !	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3x 0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x @	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x P	Q	R	S	T	U	V	W	X	Y	Z	[\	\]	X	
6x `	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x p	q	r	s	t	u	v	w	x	y	z	{	}	~	DEL	

Figura 48.1:

48.1 x64

48.1.1

Listing 48.1: Sin optimización MSVC 2013 (x64)

```
1 c$ = 8
2 toupper PROC
3     mov     BYTE PTR [rsp+8], cl
```

```

4      movsx  eax, BYTE PTR c$[rsp]
5      cmp    eax, 97
6      j1    SHORT $LN2@toupper
7      movsx  eax, BYTE PTR c$[rsp]
8      cmp    eax, 122
9      jg    SHORT $LN2@toupper
10     movsx  eax, BYTE PTR c$[rsp]
11     sub    eax, 32
12     jmp    SHORT $LN3@toupper
13     jmp    SHORT $LN1@toupper ; compiler artefact
14 $LN2@toupper:
15     movzx  eax, BYTE PTR c$[rsp] ; unnecessary casting
16 $LN1@toupper:
17 $LN3@toupper: ; compiler artefact
18     ret    0
19 toupper ENDP

```

Sin optimización GCC :

Listing 48.2: Sin optimización GCC 4.9 (x64)

```

toupper:
    push   rbp
    mov    rbp, rsp
    mov    eax, edi
    mov    BYTE PTR [rbp-4], al
    cmp    BYTE PTR [rbp-4], 96
    jle    .L2
    cmp    BYTE PTR [rbp-4], 122
    jg    .L2
    movzx  eax, BYTE PTR [rbp-4]
    sub    eax, 32
    jmp    .L3
.L2:
    movzx  eax, BYTE PTR [rbp-4]
.L3:
    pop    rbp
    ret

```

48.1.2

Con optimización MSVC :

Listing 48.3: Con optimización MSVC 2013 (x64)

toupper PROC	
lea	eax, DWORD PTR [rcx-97]

```

    cmp    al, 25
    ja     SHORT $LN2@toupper
    movsx  eax, cl
    sub    eax, 32
    ret    0
$LN2@toupper:
    movzx  eax, cl
    ret    0
toupper ENDP

```

: [42.2.1 on page 645](#).

```

int tmp=c-97;

if (tmp>25)
    return c;
else
    return c-32;

```

Con optimización GCC

Listing 48.4: Con optimización GCC 4.9 (x64)

```

1 toupper:
2     lea    edx, [rdi-97] ; 0x61
3     lea    eax, [rdi-32] ; 0x20
4     cmp    dl, 25
5     cmova eax, edi
6     ret

```

48.2 ARM

Listing 48.5: Con optimización Keil 6/2013 (Modo ARM)

```

toupper PROC
    SUB    r1,r0,#0x61
    CMP    r1,#0x19
    SUBLS r0,r0,#0x20
    ANDLS r0,r0,#0xff
    BX    lr
ENDP

```

Listing 48.6: Con optimización Keil 6/2013 (Modo Thumb)

```

toupper PROC
    MOVS   r1,r0

```

```

SUBS    r1,r1,#0x61
CMP     r1,#0x19
BHI    |L0.14|
SUBS    r0,r0,#0x20
LSLS    r0,r0,#24
LSRS    r0,r0,#24
|L0.14|
BX      lr
ENDP

```

48.2.1 GCC para ARM64

Listing 48.7: Sin optimización GCC 4.9 (ARM64)

```

toupper:
    sub    sp, sp, #16
    strb   w0, [sp,15]
    ldrb   w0, [sp,15]
    cmp    w0, 96
    bls    .L2
    ldrb   w0, [sp,15]
    cmp    w0, 122
    bhi    .L2
    ldrb   w0, [sp,15]
    sub    w0, w0, #32
    uxtb   w0, w0
    b      .L3
.L2:
    ldrb   w0, [sp,15]
.L3:
    add    sp, sp, 16
    ret

```

Listing 48.8: Con optimización GCC 4.9 (ARM64)

```

toupper:
    uxtb   w0, w0
    sub    w1, w0, #97
    uxtb   w1, w1
    cmp    w1, 25
    bhi    .L2
    sub    w0, w0, #32
    uxtb   w0, w0
.L2:
    ret

```


Capítulo 49

49.1 (x86)

:

```
add    [ebp-31F7Bh], cl
dec    dword ptr [ecx-3277Bh]
dec    dword ptr [ebp-2CF7Bh]
inc    dword ptr [ebx-7A76F33Ch]
fdiv   st(4), st
db 0FFh
dec    dword ptr [ecx-21F7Bh]
dec    dword ptr [ecx-22373h]
dec    dword ptr [ecx-2276Bh]
dec    dword ptr [ecx-22B63h]
dec    dword ptr [ecx-22F4Bh]
dec    dword ptr [ecx-23343h]
jmp    dword ptr [esi-74h]
xchg   eax, ebp
clc
std
db 0FFh
db 0FFh
mov    word ptr [ebp-214h], cs ; <-
mov    word ptr [ebp-238h], ds
mov    word ptr [ebp-23Ch], es
mov    word ptr [ebp-240h], fs
mov    word ptr [ebp-244h], gs
pushf
pop    dword ptr [ebp-210h]
```

```

mov    eax, [ebp+4]
mov    [ebp-218h], eax
lea    eax, [ebp+4]
mov    [ebp-20Ch], eax
mov    dword ptr [ebp-2D0h], 10001h
mov    eax, [eax-4]
mov    [ebp-21Ch], eax
mov    eax, [ebp+0Ch]
mov    [ebp-320h], eax
mov    eax, [ebp+10h]
mov    [ebp-31Ch], eax
mov    eax, [ebp+4]
mov    [ebp-314h], eax
call   ds:IsDebuggerPresent
mov    edi, eax
lea    eax, [ebp-328h]
push   eax
call   sub_407663
pop    ecx
test   eax, eax
jnz    short loc_402D7B

```

49.2 ?

:

- . PUSH, MOV, CALL, .
- immediates.
- .

Listing 49.1: Ruido aleatorio (x86)

```

mov    bl, 0Ch
mov    ecx, 0D38558Dh
mov    eax, ds:2C869A86h
db    67h
mov    dl, 0CCh
insb
movsb
push   eax
xor    [edx-53h], ah
fcom  qword ptr [edi-45A0EF72h]
pop    esp
pop    ss

```

```

in    eax, dx
dec   ebx
push  esp
lds   esp, [esi-41h]
retf
rcl   dword ptr [eax], cl
mov   cl, 9Ch
mov   ch, 0DFh
push  cs
insb
mov   esi, 0D9C65E4Dh
imul  ebp, [ecx], 66h
pushf
sal   dword ptr [ebp-64h], cl
sub   eax, 0AC433D64h
out   8Ch, eax
pop   ss
sbb   [eax], ebx
aas
xchg  cl, [ebx+ebx*4+14B31Eh]
jecxz short near ptr loc_58+1
xor   al, 0C6h
inc   edx
db    36h
pusha
stosb
test  [ebx], ebx
sub   al, 0D3h ; 'L'
pop   eax
stosb

loc_58: ; CODE XREF: seg000:0000004A
test  [esi], eax
inc   ebp
das
db    64h
pop   ecx
das
hlt

pop   edx
out   0B0h, al
lodsb
push  ebx
cdq
out   dx, al
sub   al, 0Ah

```

```

sti
outsd
add    dword ptr [edx], 96FCBE4Bh
and    eax, 0E537EE4Fh
inc    esp
stosd
cdq
push   ecx
in     al, 0CBh
mov    ds:0D114C45Ch, al
mov    esi, 659D1985h

```

Listing 49.2: Ruido aleatorio (x86-64)

```

lea    esi, [rax+rdx*4+43558D29h]

loc_AF3: ; CODE XREF: seg000:00000000000000B46
rcl    byte ptr [rsi+rax*8+29BB423Ah], 1
lea    ecx, cs:0FFFFFFFB2A6780Fh
mov    al, 96h
mov    ah, 0CEh
push   rsp
lodsd byte ptr [esi]

db 2Fh ; /

pop   rsp
db 64h
retf  0E993h

cmp   ah, [rax+4Ah]
movzx rsi, dword ptr [rbp-25h]
push   4Ah
movzx rdi, dword ptr [rdi+rdx*8]

db 9Ah

rcr    byte ptr [rax+1Dh], cl
lodsd
xor   [rbp+6CF20173h], edx
xor   [rbp+66F8B593h], edx
push   rbx
sbb   ch, [rbx-0Fh]
stosd
int   87h
db    46h, 4Ch
out   33h, rax
xchg  eax, ebp

```

```

test    ecx, ebp
movsd
leave
push    rsp

db 16h

xchg    eax, esi
pop     rdi

loc_B3D: ; CODE XREF: seg000:00000000000000B5F
    mov    ds:93CA685DF98A90F9h, eax
    jnz    short near ptr loc_AF3+6
    out    dx, eax
    cwde
    mov    bh, 5Dh ; ']'
    movsb
    pop    rbp

```

Listing 49.3: Ruido aleatorio (ARM (Modo ARM))

BLNE	0xFE16A9D8
BGE	0x1634D0C
SVCCS	0x450685
STRNVT	R5, [PC], #-0x964
LDCGE	p6, c14, [R0], #0x168
STCCSL	p9, c9, [LR], #0x14C
CMNHIP	PC, R10, LSL#22
FLDMIADNV	LR!, {D4}
MCR	p5, 2, R2, c15, c6, 4
BLGE	0x1139558
BLGT	0xFF9146E4
STRNEB	R5, [R4], #0xCA2
STMNEIB	R5, {R0, R4, R6, R7, R9-SP, PC}
STMIA	R8, {R0, R2-R4, R7, R8, R10, SP, LR}^
STRB	SP, [R8], PC, ROR#18
LDCCS	p9, c13, [R6, #0x1BC]
LDRGE	R8, [R9, #0x66E]
STRNEB	R5, [R8], #-0x8C3
STCCSL	p15, c9, [R7, #-0x84]
RSBLS	LR, R2, R11, ASR LR
SVCGT	0x9B0362
SVCGT	0xA73173
STMNEDB	R11!, {R0, R1, R4-R6, R8, R10, R11, SP}
STR	R0, [R3], #-0xCE4
LDCGT	p15, c8, [R1, #0x2CC]
LDRCCB	R1, [R11], -R7, ROR#30
BLLT	0xFED9D58C

BL	0x13E60F4
LDMVSIB	R3!, {R1,R4-R7}^
USATNE	R10, #7, SP, LSL#11
LDRGEB	LR, [R1],#0xE56
STRPLT	R9, [LR],#0x567
LDRLT	R11, [R1],#-0x29B
SVCNV	0x12DB29
MVNNVS	R5, SP, LSL#25
LDCL	p8, c14, [R12,#-0x288]
STCNEL	p2, c6, [R6,#-0xBC]!
SVCNV	0x2E5A2F
BLX	0x1A8C97E
TEQGE	R3, #0x1100000
STMLSIA	R6, {R3,R6,R10,R11,SP}
BICPLS	R12, R2, #0x5800
BNE	0x7CC408
TEQGE	R2, R4, LSL#20
SUBS	R1, R11, #0x28C
BICVS	R3, R12, R7,ASR R0
LDRMI	R7, [LR],R3,LSL#21
BLMI	0x1A79234
STMVCDB	R6, {R0-R3,R6,R7,R10,R11}
EORMI	R12, R6, #0xC5
MCRRCS	p1, 0xF, R1,R3,c2

Listing 49.4: Ruido aleatorio (ARM (Modo Thumb))

LSRS	R3, R6, #0x12
LDRH	R1, [R7,#0x2C]
SUBS	R0, #0x55 ; 'U'
ADR	R1, loc_3C
LDR	R2, [SP,#0x218]
CMP	R4, #0x86
SXTB	R7, R4
LDR	R4, [R1,#0x4C]
STR	R4, [R4,R2]
STR	R0, [R6,#0x20]
BGT	0xFFFFFFFF72
LDRH	R7, [R2,#0x34]
LDRSH	R0, [R2,R4]
LDRB	R2, [R7,R2]
DCB	0x17
DCB	0xED

STRB	R3, [R1,R1]
STR	R5, [R0,#0x6C]
LDMIA	R3, {R0-R5,R7}

```

ASRS    R3, R2, #3
LDR     R4, [SP,#0x2C4]
SVC     0xB5
LDR     R6, [R1,#0x40]
LDR     R5, =0xB2C5CA32
STMIA   R6, {R1-R4,R6}
LDR     R1, [R3,#0x3C]
STR     R1, [R5,#0x60]
BCC    0xFFFFFFF70
LDR     R4, [SP,#0x1D4]
STR     R5, [R5,#0x40]
ORRS   R5, R7

loc_3C ; DATA XREF: ROM:00000006
B      0xFFFFFFF98

```

Listing 49.5: Ruido aleatorio (MIPS little endian)

```

lw      $t9, 0xCB3($t5)
sb      $t5, 0x3855($t0)
sltiu  $a2, $a0, -0x657A
ldr    $t4, -0x4D99($a2)
daddi  $s0, $s1, 0x50A4
lw      $s7, -0x2353($s4)
bgtzl $a1, 0x17C5C

.byte 0x17
.byte 0xED
.byte 0x4B  # K
.byte 0x54  # T

lwc2   $31, 0x66C5($sp)
lwu    $s1, 0x10D3($a1)
ldr    $t6, -0x204B($zero)
lwc1   $f30, 0x4DBE($s2)
daddiu $t1, $s1, 0x6BD9
lwu    $s5, -0x2C64($v1)
cop0   0x13D642D
bne   $gp, $t4, 0xFFFF9EF0
lh    $ra, 0x1819($s1)
sdl   $fp, -0x6474($t8)
jal   0x78C0050
ori   $v0, $s2, 0xC634
blez  $gp, 0xFFFFEA9D4
swl   $t8, -0x2CD4($s2)
sltiu $a1, $k0, 0x685
sdc1   $f15, 0x5964($at)
sw    $s0, -0x19A6($a1)

```

sltiu	\$t6, \$a3, -0x66AD
lb	\$t7, -0x4F6(\$t3)
sd	\$fp, 0x4B02(\$a1)

Capítulo 50

50.1

([57 on page 850](#)) . . .

```
:  
mov    byte ptr [ebx], 'h'  
mov    byte ptr [ebx+1], 'e'  
mov    byte ptr [ebx+2], 'l'  
mov    byte ptr [ebx+3], 'l'  
mov    byte ptr [ebx+4], 'o'  
mov    byte ptr [ebx+5], ' '  
mov    byte ptr [ebx+6], 'w'  
mov    byte ptr [ebx+7], 'o'  
mov    byte ptr [ebx+8], 'r'  
mov    byte ptr [ebx+9], 'l'  
mov    byte ptr [ebx+10], 'd'
```

```
:  
mov    ebx, offset username  
cmp    byte ptr [ebx], 'j'  
jnz    fail  
cmp    byte ptr [ebx+1], 'o'  
jnz    fail  
cmp    byte ptr [ebx+2], 'h'  
jnz    fail  
cmp    byte ptr [ebx+3], 'n'  
jnz    fail
```

```
    jz      it_is_john
```

sprintf():

```
sprintf(buf, "%s%c%s%c%s", "hel",'l',"o w",'o','rld');
```

[: 78.2 on page 986.](#)

50.2

50.2.1

.

:

Listing 50.1:

```
add    eax, ebx
mul    ecx
```

Listing 50.2: obfuscated code

```
xor    esi, 011223344h ;
add    esi, eax        ;
add    eax, ebx        ;
mov    edx, eax        ;
shl    edx, 4          ;
mul    ecx             ;
xor    esi, ecx        ;
```

(ESI y EDX).?

50.2.2

- MOV op1, op2 PUSH op2 / POP op1.
- JMP label PUSH label / RET. [IDA](#).
- CALL label:PUSH label_after_CALL_instruction / PUSH label / RET.
- PUSH op:SUB ESP, 4 (o 8) / MOV [ESP], op.

50.2.3

ESI :

```
mov    esi, 1
...
dec    esi
...
cmp    esi, 0
jz     real_code
;
real_code:
```

everse engineer.

opaque predicate.

(ESI):

```
add    eax, ebx      ;
mul    ecx          ;
add    eax, esi      ; opaque predicate.
```

50.2.4

```
instruction 1
instruction 2
instruction 3
```

:

```
begin:        jmp    ins1_label
ins2_label:   instruction 2
              jmp    ins3_label
ins3_label:   instruction 3
              jmp    exit:
ins1_label:   instruction 1
              jmp    ins2_label
exit:
```

50.2.5

```

dummy_data1    db      100h dup (0)
message1       db      'hello world',0

dummy_data2    db      200h dup (0)
message2       db      'another message',0

func           proc
...
    mov     eax, offset dummy_data1 ; PE or ELF ↵
↳ reloc here
    add     eax, 100h
    push    eax
    call    dump_string
...
    mov     eax, offset dummy_data2 ; PE or ELF ↵
↳ reloc here
    add     eax, 200h
    push    eax
    call    dump_string
...
func           endp

```

[IDA](#) dummy_data1 y dummy_data2, .

50.3

PL o ISA . (.NET or Java machines). .

50.4

:<http://go.yurichev.com/17220>.

MOV : [Dol13].

50.5 Ejercicios

50.5.1 Ejercicio #1

¹ .

¹blog.yurichev.com

beginners.re.

: G.1.15 on page 1267.

Capítulo 51

C++

51.1

51.1.1

```
#include <stdio.h>

class c
{
private:
    int v1;
    int v2;
public:
    c() // 
    {
        v1=667;
        v2=999;
    }

    c(int a, int b) // 
    {
        v1=a;
        v2=b;
    }

    void dump()
    {
        printf ("%d; %d\n", v1, v2);
    }
}
```

```

};

int main()
{
    class c c1;
    class c c2(5,6);

    c1.dump();
    c2.dump();

    return 0;
}

```

MSVCx86

Listing 51.1: MSVC

```

_c2$ = -16 ; size = 8
_c1$ = -8 ; size = 8
_main PROC
    push ebp
    mov  ebp, esp
    sub  esp, 16
    lea   ecx, DWORD PTR _c1$[ebp]
    call ??0c@@QAE@XZ ; c::c
    push 6
    push 5
    lea   ecx, DWORD PTR _c2$[ebp]
    call ??0c@@QAE@HH@Z ; c::c
    lea   ecx, DWORD PTR _c1$[ebp]
    call ?dump@c@@QAEXXZ ; c::dump
    lea   ecx, DWORD PTR _c2$[ebp]
    call ?dump@c@@QAEXXZ ; c::dump
    xor  eax, eax
    mov  esp, ebp
    pop  ebp
    ret  0
_main ENDP

```

Listing 51.2: MSVC

```

this$ = -4          ; size = 4
??0c@@QAE@XZ PROC ; c::c, COMDAT
; _this$ = ecx
    push ebp
    mov  ebp, esp
    push ecx

```

```

    mov  DWORD PTR _this$[ebp], ecx
    mov  eax, DWORD PTR _this$[ebp]
    mov  DWORD PTR [eax], 667
    mov  ecx, DWORD PTR _this$[ebp]
    mov  DWORD PTR [ecx+4], 999
    mov  eax, DWORD PTR _this$[ebp]
    mov  esp, ebp
    pop  ebp
    ret  0
??0c@@QAE@XZ ENDP ; c::c

_this$ = -4 ; size = 4
_a$ = 8      ; size = 4
_b$ = 12     ; size = 4
??0c@@QAE@H@Z PROC ; c::c, COMDAT
; _this$ = ecx
    push ebp
    mov  ebp, esp
    push ecx
    mov  DWORD PTR _this$[ebp], ecx
    mov  eax, DWORD PTR _this$[ebp]
    mov  ecx, DWORD PTR _a$[ebp]
    mov  DWORD PTR [eax], ecx
    mov  edx, DWORD PTR _this$[ebp]
    mov  eax, DWORD PTR _b$[ebp]
    mov  DWORD PTR [edx+4], eax
    mov  eax, DWORD PTR _this$[ebp]
    mov  esp, ebp
    pop  ebp
    ret  8
??0c@@QAE@H@Z ENDP ; c::c

```

[ISO13, pág. 12.1] . , *this*.

Listing 51.3: MSVC

```

_this$ = -4          ; size = 4
?dump@c@@QAEXXZ PROC ; c::dump, COMDAT
; _this$ = ecx
    push ebp
    mov  ebp, esp
    push ecx
    mov  DWORD PTR _this$[ebp], ecx
    mov  eax, DWORD PTR _this$[ebp]
    mov  ecx, DWORD PTR [eax+4]
    push ecx
    mov  edx, DWORD PTR _this$[ebp]
    mov  eax, DWORD PTR [edx]

```

```

push eax
push OFFSET ??_C@_07NJBDCIEC@?$CFd?$DL?5?$CFd?6?$AA@
call _printf
add esp, 12
mov esp, ebp
pop ebp
ret 0
?dump@c@@QAEXXZ ENDP ; c::dump

```

Listing 51.4: MSVC

```

??0c@@QAE@XZ PROC ; c::c, COMDAT
; _this$ = ecx
    mov eax, ecx
    mov DWORD PTR [eax], 667
    mov DWORD PTR [eax+4], 999
    ret 0
??0c@@QAE@XZ ENDP ; c::c

_a$ = 8 ; size = 4
_b$ = 12 ; size = 4
??0c@@QAE@HH@Z PROC ; c::c, COMDAT
; _this$ = ecx
    mov edx, DWORD PTR _b$[esp-4]
    mov eax, ecx
    mov ecx, DWORD PTR _a$[esp-4]
    mov DWORD PTR [eax], ecx
    mov DWORD PTR [eax+4], edx
    ret 8
??0c@@QAE@HH@Z ENDP ; c::c

?dump@c@@QAEXXZ PROC ; c::dump, COMDAT
; _this$ = ecx
    mov eax, DWORD PTR [ecx+4]
    mov ecx, DWORD PTR [ecx]
    push eax
    push ecx
    push OFFSET ??_C@_07NJBDCIEC@?$CFd?$DL?5?$CFd?6?$AA@
    call _printf
    add esp, 12
    ret 0
?dump@c@@QAEXXZ ENDP ; c::dump

```

(64 on page 869).

MSVCx86-64

. Spanish text placeholder. c(int a, int b):

Listing 51.5: Con optimización MSVC 2012 x64

```
; void dump()

?dump@c@@QEAXXXZ PROC ; c::dump
    mov    r8d, DWORD PTR [rcx+4]
    mov    edx, DWORD PTR [rcx]
    lea    rcx, OFFSET FLAT:??_C@_07NJBDCIEC@?$CFd?$DL?5?$CFd@
        ↴ ?6?$AA@ ; '%d; %d'
    jmp    printf
?dump@c@@QEAXXXZ ENDP ; c::dump

; c(int a, int b)

??0c@@QEAA@HH@Z PROC ; c::c
    mov    DWORD PTR [rcx], edx ; : a
    mov    DWORD PTR [rcx+4], r8d ; : b
    mov    rax, rcx
    ret    0
??0c@@QEAA@HH@Z ENDP ; c::c

;

??0c@@QEAA@XZ PROC ; c::c
    mov    DWORD PTR [rcx], 667
    mov    DWORD PTR [rcx+4], 999
    mov    rax, rcx
    ret    0
??0c@@QEAA@XZ ENDP ; c::c
```

: 13.1.1 on page 184.

GCCx86

Listing 51.6: GCC 4.4.1

```
public main
main proc near

var_20 = dword ptr -20h
var_1C = dword ptr -1Ch
var_18 = dword ptr -18h
```

```

var_10 = dword ptr -10h
var_8  = dword ptr -8

push ebp
mov  ebp, esp
and  esp, 0FFFFFFF0h
sub  esp, 20h
lea   eax, [esp+20h+var_8]
mov  [esp+20h+var_20], eax
call _ZN1cC1Ev
mov  [esp+20h+var_18], 6
mov  [esp+20h+var_1C], 5
lea   eax, [esp+20h+var_10]
mov  [esp+20h+var_20], eax
call _ZN1cC1Eii
lea   eax, [esp+20h+var_8]
mov  [esp+20h+var_20], eax
call _ZN1c4dumpEv
lea   eax, [esp+20h+var_10]
mov  [esp+20h+var_20], eax
call _ZN1c4dumpEv
mov  eax, 0
leave
retn
main endp

```

```

_ZN1cC1Ev    public _ZN1cC1Ev ; weak
               proc near           ; CODE XREF: main+10
arg_0         = dword ptr  8

               push    ebp
               mov     ebp, esp
               mov     eax, [ebp+arg_0]
               mov     dword ptr [eax], 667
               mov     eax, [ebp+arg_0]
               mov     dword ptr [eax+4], 999
               pop    ebp
               retn
_ZN1cC1Ev    endp

```

```

_ZN1cC1Eii    public _ZN1cC1Eii
               proc near

arg_0         = dword ptr  8
arg_4         = dword ptr  0Ch
arg_8         = dword ptr  10h

```

```

        push    ebp
        mov     ebp, esp
        mov     eax, [ebp+arg_0]
        mov     edx, [ebp+arg_4]
        mov     [eax], edx
        mov     eax, [ebp+arg_0]
        mov     edx, [ebp+arg_8]
        mov     [eax+4], edx
        pop    ebp
        retn
_ZN1cC1Eii      endp

```

```

void ZN1cC1Eii (int *obj, int a, int b)
{
    *obj=a;
    *(obj+1)=b;
}

```

....

```

_ZN1c4dumpEv    public _ZN1c4dumpEv
                 proc near

var_18           = dword ptr -18h
var_14           = dword ptr -14h
var_10           = dword ptr -10h
arg_0            = dword ptr  8

        push    ebp
        mov     ebp, esp
        sub    esp, 18h
        mov     eax, [ebp+arg_0]
        mov     edx, [eax+4]
        mov     eax, [ebp+arg_0]
        mov     eax, [eax]
        mov     [esp+18h+var_10], edx
        mov     [esp+18h+var_14], eax
        mov     [esp+18h+var_18], offset aDD ; "%d\"
        ↳ n"
        call    _printf
        leave
        retn
_ZN1c4dumpEv    endp

```

:

```
void ZN1c4dumpEv (int *obj)
{
    printf ("%d; %d\n", *obj, *(obj+1));
}
```

GCCx86-64

rdi, rsi, rdx, rcx, r8 and r9 [Mit13]... .

Listing 51.7: GCC 4.4.6 x64

```
;

_ZN1cC2Ev:
    mov    DWORD PTR [rdi], 667
    mov    DWORD PTR [rdi+4], 999
    ret

; c(int a, int b)

_ZN1cC2Eii:
    mov    DWORD PTR [rdi], esi
    mov    DWORD PTR [rdi+4], edx
    ret

; dump()

_ZN1c4dumpEv:
    mov    edx, DWORD PTR [rdi+4]
    mov    esi, DWORD PTR [rdi]
    xor    eax, eax
    mov    edi, OFFSET FLAT:.LC0 ; "%d; %d\n"
    jmp    printf
```

51.1.2

:

```
#include <stdio.h>

class object
{
    public:
        int color;
        object() { };
```

```
        object (int color) { this->color=color; };
        void print_color() { printf ("color=%d\n", color); };
};

class box : public object
{
    private:
        int width, height, depth;
    public:
        box(int color, int width, int height, int depth)
        {
            this->color=color;
            this->width=width;
            this->height=height;
            this->depth=depth;
        };
        void dump()
        {
            printf ("this is box. color=%d, width=%d, height=%d,
        ↴ , depth=%d\n", color, width, height, depth);
        };
};

class sphere : public object
{
private:
    int radius;
public:
    sphere(int color, int radius)
    {
        this->color=color;
        this->radius=radius;
    };
    void dump()
    {
        printf ("this is sphere. color=%d, radius=%d\n", color,
        ↴ radius);
    };
};

int main()
{
    box b(1, 10, 20, 30);
    sphere s(2, 40);

    b.print_color();
    s.print_color();
```

```

    b.dump();
    s.dump();

    return 0;
}

```

1

Listing 51.8: Con optimización MSVC 2008 /Ob0

```

??_C@_09GCEDOLPA@color?$DN?$CFd?6?$AA@ DB 'color=%d', 0aH, 00H ↵
    ↴ ; `string'
?print_color@object@@QAEXXZ PROC ; object::print_color, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx]
    push eax

; 'color=%d', 0aH, 00H
    push OFFSET ??_C@_09GCEDOLPA@color?$DN?$CFd?6?$AA@
    call _printf
    add  esp, 8
    ret  0
?print_color@object@@QAEXXZ ENDP ; object::print_color

```

Listing 51.9: Con optimización MSVC 2008 /Ob0

```

?dump@box@@QAEXXZ PROC ; box::dump, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx+12]
    mov  edx, DWORD PTR [ecx+8]
    push eax
    mov  eax, DWORD PTR [ecx+4]
    mov  ecx, DWORD PTR [ecx]
    push edx
    push eax
    push ecx

; 'this is box. color=%d, width=%d, height=%d, depth=%d', 0aH, ↵
    ↴ 00H ; `string'
    push OFFSET ??_C@_0DG@NCNGAADL@this?5is?5box?4?5color?$DN?↵
    ↴ $CFd?0?5width?$DN?$CFd?0@
    call _printf
    add  esp, 20
    ret  0
?dump@box@@QAEXXZ ENDP ; box::dump

```

Listing 51.10: Con optimización MSVC 2008 /Ob0

```
?dump@sphere@@QAEXXZ PROC ; sphere::dump, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx+4]
    mov  ecx, DWORD PTR [ecx]
    push eax
    push ecx

; 'this is sphere. color=%d, radius=%d', 0aH, 00H
    push OFFSET ??_C@_0CF@EFEDJLDC@this?5is?5sphere?4?5color?2
    \ $DN?$CFd?0?5radius@
    call _printf
    add  esp, 12
    ret  0
?dump@sphere@@QAEXXZ ENDP ; sphere::dump
```

:

Spanish text placeholder	Spanish text placeholder
+0x0	int color

box:

Spanish text placeholder	Spanish text placeholder
+0x0	int color
+0x4	int width
+0x8	int height
+0xC	int depth

sphere:

Spanish text placeholder	Spanish text placeholder
+0x0	int color
+0x4	int radius

:

Listing 51.11: Con optimización MSVC 2008 /Ob0

```
PUBLIC _main
_TEXT SEGMENT
_s$ = -24 ; size = 8
_b$ = -16 ; size = 16
_main PROC
    sub esp, 24
    push 30
    push 20
    push 10
    push 1
```

```

    lea    ecx, DWORD PTR _b$[esp+40]
    call  ??0box@@QAE@HHHH@Z ; box::box
    push  40
    push  2
    lea    ecx, DWORD PTR _s$[esp+32]
    call  ??0sphere@@QAE@HH@Z ; sphere::sphere
    lea    ecx, DWORD PTR _b$[esp+24]
    call  ?print_color@object@@QAEXXZ ; object::print_color
    lea    ecx, DWORD PTR _s$[esp+24]
    call  ?print_color@object@@QAEXXZ ; object::print_color
    lea    ecx, DWORD PTR _b$[esp+24]
    call  ?dump@box@@QAEXXZ ; box::dump
    lea    ecx, DWORD PTR _s$[esp+24]
    call  ?dump@sphere@@QAEXXZ ; sphere::dump
    xor   eax, eax
    add   esp, 24
    ret   0
_main ENDP

```

51.1.3

```

#include <stdio.h>

class box
{
private:
    int color, width, height, depth;
public:
    box(int color, int width, int height, int depth)
    {
        this->color=color;
        this->width=width;
        this->height=height;
        this->depth=depth;
    };
    void dump()
    {
        printf ("this is box. color=%d, width=%d, height=%d,
        , depth=%d\n", color, width, height, depth);
    };
};

```

```

?dump@box@@QAEXXZ PROC ; box::dump, COMDAT
; _this$ = ecx
    mov   eax, DWORD PTR [ecx+12]
    mov   edx, DWORD PTR [ecx+8]

```

```

push eax
mov eax, DWORD PTR [ecx+4]
mov ecx, DWORD PTR [ecx]
push edx
push eax
push ecx
; 'this is box. color=%d, width=%d, height=%d, depth=%d', 0aH, ↴
↳ 00H
push OFFSET ??_C@_0DG@NCNGAADL@this?5is?5box?4?5color?$DN?2
↳ $CFd?0?5width?$DN?$CFd?0@
call _printf
add esp, 20
ret 0
?dump@box@@QAEXXZ ENDP ; box::dump

```

Spanish text placeholder	Spanish text placeholder
+0x0	int color
+0x4	int width
+0x8	int height
+0xC	int depth

```

void hack_oop_encapsulation(class box * o)
{
    o->width=1; // :
                // "error C2248: 'box::width' : cannot access ↴
    ↳ private member declared in class 'box'"
};

```

```

void hack_oop_encapsulation(class box * o)
{
    unsigned int *ptr_to_object=reinterpret_cast<unsigned int>(
        *(o));
    ptr_to_object[1]=123;
};

```

```

?hack_oop_encapsulation@@YAXPAVbox@@@Z PROC ; ↴
↳ hack_oop_encapsulation
    mov eax, DWORD PTR _o$[esp-4]
    mov DWORD PTR [eax+4], 123
    ret 0
?hack_oop_encapsulation@@YAXPAVbox@@@Z ENDP ; ↴
↳ hack_oop_encapsulation

```

```

int main()
{

```

```

    box b(1, 10, 20, 30);

    b.dump();

    hack_oop_encapsulation(&b);

    b.dump();

    return 0;
}

```

```

this is box. color=1, width=10, height=20, depth=30
this is box. color=1, width=123, height=20, depth=30

```

51.1.4

```

#include <stdio.h>

class box
{
public:
    int width, height, depth;
    box() { };
    box(int width, int height, int depth)
    {
        this->width=width;
        this->height=height;
        this->depth=depth;
    };
    void dump()
    {
        printf ("this is box. width=%d, height=%d, depth=%d\n",
        width, height, depth);
    };
    int get_volume()
    {
        return width * height * depth;
    };
};

class solid_object
{
public:
    int density;
    solid_object() { };
    solid_object(int density)

```

```
{      this->density=density;
};
int get_density()
{
    return density;
};
void dump()
{
    printf ("this is solid_object. density=%d\n", ↴
↳ density);
};

class solid_box: box, solid_object
{
public:
    solid_box (int width, int height, int depth, int ↴
↳ density)
    {
        this->width=width;
        this->height=height;
        this->depth=depth;
        this->density=density;
    };
    void dump()
    {
        printf ("this is solid_box. width=%d, height=%d, ↴
↳ depth=%d, density=%d\n", width, height, depth, density);
    };
    int get_weight() { return get_volume() * get_density(); ↴
    };
};

int main()
{
    box b(10, 20, 30);
    solid_object so(100);
    solid_box sb(10, 20, 30, 3);

    b.dump();
    so.dump();
    sb.dump();
    printf ("%d\n", sb.get_weight());

    return 0;
};
```

Listing 51.12: Con optimización MSVC 2008 /Ob0

```
?dump@box@@QAEXXZ PROC ; box::dump, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx+8]
    mov  edx, DWORD PTR [ecx+4]
    push eax
    mov  eax, DWORD PTR [ecx]
    push edx
    push eax
; 'this is box. width=%d, height=%d, depth=%d', 0aH, 00H
    push OFFSET ??_C@_0CM@DIKPHDFI@this?5is?5box?4?5width?$DN?⤦
    ↴ $CFd?0?5height?$DN?$CFd@
    call _printf
    add  esp, 16
    ret  0
?dump@box@@QAEXXZ ENDP ; box::dump
```

Listing 51.13: Con optimización MSVC 2008 /Ob0

```
?dump@solid_object@@QAEXXZ PROC ; solid_object::dump, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx]
    push eax
; 'this is solid_object. density=%d', 0aH
    push OFFSET ??_C@_0CC@KICFJINL@this?5is?5solid_object?4?5⤦
    ↴ density?$DN?$CFd@
    call _printf
    add  esp, 8
    ret  0
?dump@solid_object@@QAEXXZ ENDP ; solid_object::dump
```

Listing 51.14: Con optimización MSVC 2008 /Ob0

```
?dump@solid_box@@QAEXXZ PROC ; solid_box::dump, COMDAT
; _this$ = ecx
    mov  eax, DWORD PTR [ecx+12]
    mov  edx, DWORD PTR [ecx+8]
    push eax
    mov  eax, DWORD PTR [ecx+4]
    mov  ecx, DWORD PTR [ecx]
    push edx
    push eax
    push ecx
; 'this is solid_box. width=%d, height=%d, depth=%d, density=%d',
    ↴ ', 0aH
    push OFFSET ??_C@_0DO@HNCNIHNN@this?5is?5solid_box?4?5width?⤦
    ↴ ?$DN?$CFd?0?5hei@
```

```
call _printf
add esp, 20
ret 0
?dump@solid_box@@QAEXXZ ENDP ; solid_box::dump
```

:

Spanish text placeholder	Spanish text placeholder
+0x0	width
+0x4	height
+0x8	depth

:

Spanish text placeholder	Spanish text placeholder
+0x0	density

:

Spanish text placeholder	Spanish text placeholder
+0x0	width
+0x4	height
+0x8	depth
+0xC	density

Listing 51.15: Con optimización MSVC 2008 /Ob0

```
?get_volume@box@@QAEHXZ PROC ; box::get_volume, COMDAT
; _this$ = ecx
    mov eax, DWORD PTR [ecx+8]
    imul eax, DWORD PTR [ecx+4]
    imul eax, DWORD PTR [ecx]
    ret 0
?get_volume@box@@QAEHXZ ENDP ; box::get_volume
```

Listing 51.16: Con optimización MSVC 2008 /Ob0

```
?get_density@solid_object@@QAEHXZ PROC ; solid_object::`v
    ↳ get_density, COMDAT
; _this$ = ecx
    mov eax, DWORD PTR [ecx]
    ret 0
?get_density@solid_object@@QAEHXZ ENDP ; solid_object::`v
    ↳ get_density
```

Listing 51.17: Con optimización MSVC 2008 /Ob0

```
?get_weight@solid_box@@QAEHXZ PROC ; solid_box::get_weight, ↳
    ↳ COMDAT
```

```
; _this$ = ecx
    push esi
    mov  esi, ecx
    push edi
    lea  ecx, DWORD PTR [esi+12]
    call ?get_density@solid_object@@QAEHXZ ; solid_object::get_density
    mov  eax, edi
    mov  edi, eax
    call ?get_volume@box@@QAEHXZ ; box::get_volume
    imul eax, edi
    pop  edi
    pop  esi
    ret  0
?get_weight@solid_box@@QAEHXZ ENDP ; solid_box::get_weight
```

51.1.5

```
#include <stdio.h>

class object
{
public:
    int color;
    object() { };
    object (int color) { this->color=color; };
    virtual void dump()
    {
        printf ("color=%d\n", color);
    };
};

class box : public object
{
private:
    int width, height, depth;
public:
    box(int color, int width, int height, int depth)
    {
        this->color=color;
        this->width=width;
        this->height=height;
        this->depth=depth;
    };
    void dump()
    {
```

```

        printf ("this is box. color=%d, width=%d, height=%d\n",
        ↴ , depth=%d\n", color, width, height, depth);
    };
};

class sphere : public object
{
    private:
        int radius;
    public:
        sphere(int color, int radius)
    {
        this->color=color;
        this->radius=radius;
    };
    void dump()
    {
        printf ("this is sphere. color=%d, radius=%d\n",
        ↴ color, radius);
    };
};

int main()
{
    box b(1, 10, 20, 30);
    sphere s(2, 40);

    object *o1=&b;
    object *o2=&s;

    o1->dump();
    o2->dump();
    return 0;
};

```

```

_s$ = -32 ; size = 12
_b$ = -20 ; size = 20
_main PROC
    sub esp, 32
    push 30
    push 20
    push 10
    push 1
    lea ecx, DWORD PTR _b$[esp+48]
    call ??0box@@QAE@HHHH@Z ; box::box
    push 40
    push 2

```

```

    lea    ecx, DWORD PTR _s$[esp+40]
    call  ??0sphere@@QAE@HH@Z ; sphere::sphere
    mov    eax, DWORD PTR _b$[esp+32]
    mov    edx, DWORD PTR [eax]
    lea    ecx, DWORD PTR _b$[esp+32]
    call  edx
    mov    eax, DWORD PTR _s$[esp+32]
    mov    edx, DWORD PTR [eax]
    lea    ecx, DWORD PTR _s$[esp+32]
    call  edx
    xor    eax, eax
    add    esp, 32
    ret    0
_main ENDP

```

[2](#)

```

??_R0?AVbox@@@8 DD FLAT:??_7type_info@@@6B@ ; box `RTTI Type ↵
↳ Descriptor'
DD    00H
DB    '.?AVbox@@', 00H

??_R1A@?0A@EA@box@@8 DD FLAT:??_R0?AVbox@@@8 ; box::`RTTI Base ↵
↳ Class Descriptor at (0,-1,0,64)'
DD    01H
DD    00H
DD    0xffffffffH
DD    00H
DD    040H
DD    FLAT:??_R3box@@8

??_R2box@@8 DD     FLAT:??_R1A@?0A@EA@box@@8 ; box::`RTTI Base ↵
↳ Class Array'
DD    FLAT:??_R1A@?0A@EA@object@@8

??_R3box@@8 DD     00H ; box::`RTTI Class Hierarchy Descriptor'
DD    00H
DD    02H
DD    FLAT:??_R2box@@8

??_R4box@@6B@ DD 00H ; box::`RTTI Complete Object Locator'
DD    00H
DD    00H
DD    FLAT:??_R0?AVbox@@@8
DD    FLAT:??_R3box@@8

```

²:[\(23 on page 504\)](#)

```
??_7box@@6B@ DD      FLAT:??_R4box@@6B@ ; box::`vftable'
    DD      FLAT:?dump@box@@UAEXXZ

_color$ = 8    ; size = 4
_width$ = 12   ; size = 4
_height$ = 16  ; size = 4
_depth$ = 20   ; size = 4
??0box@@QAE@HHHH@Z PROC ; box::box, COMDAT
; _this$ = ecx
    push esi
    mov  esi, ecx
    call ??0object@@QAE@XZ ; object::object
    mov  eax, DWORD PTR _color$[esp]
    mov  ecx, DWORD PTR _width$[esp]
    mov  edx, DWORD PTR _height$[esp]
    mov  DWORD PTR [esi+4], eax
    mov  eax, DWORD PTR _depth$[esp]
    mov  DWORD PTR [esi+16], eax
    mov  DWORD PTR [esi], OFFSET ??_7box@@6B@
    mov  DWORD PTR [esi+8], ecx
    mov  DWORD PTR [esi+12], edx
    mov  eax, esi
    pop  esi
    ret  16
??0box@@QAE@HHHH@Z ENDP ; box::box
```

51.2 ostream

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
}
```

ostream.:

Listing 51.18: MSVC 2012 (reduced listing)

```
$SG37112 DB 'Hello, world!', 0aH, 00H
_main PROC
    push OFFSET $SG37112
    push OFFSET ?cout@std@@3V?$basic_ostream@DU??
        ↴ $char_traits@D@std@@@1@A ; std::cout
```

```

call ??$?6U?$char_traits@D@std@@@std@@YAAV? ↵
↳ $basic_ostream@DU?$char_traits@D@std@@@0@AAV10@PBD@Z ; ↵
↳ std::operator<<<std::char_traits<char> >
add esp, 8
xor eax, eax
ret 0
_main ENDP

```

:

```

#include <iostream>

int main()
{
    std::cout << "Hello, " << "world!\n";
}

```

:

Listing 51.19: MSVC 2012

```

$SG37112 DB 'world!', 0aH, 00H
$SG37113 DB 'Hello, ', 00H

_main PROC
    push OFFSET $SG37113 ; 'Hello, '
    push OFFSET ?cout@std@@3V?$basic_ostream@DU? ↵
    ↳ $char_traits@D@std@@@1@A ; std::cout
    call ??$?6U?$char_traits@D@std@@@std@@YAAV? ↵
    ↳ $basic_ostream@DU?$char_traits@D@std@@@0@AAV10@PBD@Z ; ↵
    ↳ std::operator<<<std::char_traits<char> >
    add esp, 8

    push OFFSET $SG37112 ; 'world!'
    push eax
    call ??$?6U?$char_traits@D@std@@@std@@YAAV? ↵
    ↳ $basic_ostream@DU?$char_traits@D@std@@@0@AAV10@PBD@Z ; ↵
    ↳ std::operator<<<std::char_traits<char> >
    add esp, 8

    xor eax, eax
    ret 0
_main ENDP

```

```
f(f(std::cout, "Hello, "), "world!");
```

GCC MSVC.

51.3 References

[ISO13, pág. 8.3.2]. [Cli, pág. 8.6]. [Cli, pág. 8.5].

```
void f2 (int x, int y, int & sum, int & product)
{
    sum=x+y;
    product=x*y;
}
```

Listing 51.20: Con optimización MSVC 2010

```
_x$ = 8 ; size ↗
    ↓ = 4
_y$ = 12 ; size ↗
    ↓ = 4
_sum$ = 16 ; size ↗
    ↓ = 4
_product$ = 20 ; size ↗
    ↓ = 4
?f2@@YAXHAAH0@Z PROC ; f2
    mov     ecx, DWORD PTR _y$[esp-4]
    mov     eax, DWORD PTR _x$[esp-4]
    lea     edx, DWORD PTR [eax+ecx]
    imul   eax, ecx
    mov     ecx, DWORD PTR _product$[esp-4]
    push   esi
    mov     esi, DWORD PTR _sum$[esp]
    mov     DWORD PTR [esi], edx
    mov     DWORD PTR [ecx], eax
    pop    esi
    ret    0
?f2@@YAXHAAH0@Z ENDP ; f2
```

(: 51.1.1 on page 702.)

51.4 STL

N.B.: ...

51.4.1 std::string

MSVC

$16 + 4 + 4 = 24$ $16 + 8 + 8 = 32$.

Listing 51.21: MSVC

```
#include <string>
#include <stdio.h>

struct std_string
{
    union
    {
        char buf[16];
        char* ptr;
    } u;
    size_t size;      // AKA 'Mysize'  MSVC
    size_t capacity; // AKA 'Myres'   MSVC
};

void dump_std_string(std::string s)
{
    struct std_string *p=(struct std_string*)&s;
    printf ("[%s] size:%d capacity:%d\n", p->size>16 ? p->u.ptr:
    ↴ : p->u.buf, p->size, p->capacity);
};

int main()
{
    std::string s1="short string";
    std::string s2="string longer than 16 bytes";

    dump_std_string(s1);
    dump_std_string(s2);

    // c_str()
    printf ("%s\n", &s1);
    printf ("%s\n", s2);
};

:
```

.

.

.

.printf()

GCC

.. libstdc++-v3\include\bits\basic_string.h :

```
* The reason you want _M_data pointing to the character %>
 * array and
 * not the _Rep is so that the debugger can see the string
 * contents. (Probably we should add a non-inline member to %
 * get
 * the _Rep for the debugger to use, so users can check the %
 * actual
 * string length.)
```

[basic_string.h](#)

:

Listing 51.22: GCC

```
#include <string>
#include <stdio.h>

struct std_string
{
    size_t length;
    size_t capacity;
    size_t refcount;
};

void dump_std_string(std::string s)
{
    char *p1=*(char**)&s; //
    struct std_string *p2=(struct std_string*)(p1-offsetof(struct,
        std_string));
    printf ("[%s] size:%d capacity:%d\n", p1, p2->length, p2->%
        capacity);
};

int main()
{
    std::string s1="short string";
    std::string s2="string longer that 16 bytes";
```

```

    dump_std_string(s1);
    dump_std_string(s2);

    // :
    printf ("%s\n", *(char**)&s1);
    printf ("%s\n", *(char**)&s2);
};

.
.
```

```

#include <string>
#include <stdio.h>

int main()
{
    std::string s1="Hello, ";
    std::string s2="world!\n";
    std::string s3=s1+s2;

    printf ("%s\n", s3.c_str());
}

```

Listing 51.23: MSVC 2012

```

$SG39512 DB 'Hello, ', 00H
$SG39514 DB 'world!', 0aH, 00H
$SG39581 DB '%s', 0aH, 00H

_s2$ = -72 ; size = 24
_s3$ = -48 ; size = 24
_s1$ = -24 ; size = 24
_main PROC
    sub esp, 72

    push 7
    push OFFSET $SG39512
    lea ecx, DWORD PTR _s1$[esp+80]
    mov DWORD PTR _s1$[esp+100], 15
    mov DWORD PTR _s1$[esp+96], 0
    mov BYTE PTR _s1$[esp+80], 0
    call ?assign@?$basic_string@DU?$char_traits@D@std@@V?@
    ↴ $allocator@D@2@std@@QAEAAV12@PBDI@Z ; std::basic_string<char,
    ↴ std::char_traits<char>, std::allocator<char> >::assign
    ↴

```

```

push 7
push OFFSET $SG39514
lea ecx, DWORD PTR _s2$[esp+80]
mov DWORD PTR _s2$[esp+100], 15
mov DWORD PTR _s2$[esp+96], 0
mov BYTE PTR _s2$[esp+80], 0
call ?assign@?$basic_string@DU?$char_traits@D@std@@V?@
    ↴ $allocator@D@2@std@@QAEAAV12@PBDI@Z ; std::basic_string<,
    ↴ char, std::char_traits<char>, std::allocator<char> >::,
    ↴ assign

lea eax, DWORD PTR _s2$[esp+72]
push eax
lea eax, DWORD PTR _s1$[esp+76]
push eax
lea eax, DWORD PTR _s3$[esp+80]
push eax
call ??$?HDU?$char_traits@D@std@@V?$$allocator@D@1@@std@@YA??
    ↴ AV?$basic_string@DU?$char_traits@D@std@@V??
    ↴ $allocator@D@2@0@0@ABV10@0@Z ; std::operator+<char, std::
    ↴ char_traits<char>, std::allocator<char> >

; :
cmp DWORD PTR _s3$[esp+104], 16
lea eax, DWORD PTR _s3$[esp+84]
cmovae eax, DWORD PTR _s3$[esp+84]

push eax
push OFFSET $SG39581
call _printf
add esp, 20

cmp DWORD PTR _s3$[esp+92], 16
jb SHORT $LN119@main
push DWORD PTR _s3$[esp+72]
call ??3@YAXPAX@Z           ; operator delete
add esp, 4

$LN119@main:
cmp DWORD PTR _s2$[esp+92], 16
mov DWORD PTR _s3$[esp+92], 15
mov DWORD PTR _s3$[esp+88], 0
mov BYTE PTR _s3$[esp+72], 0
jb SHORT $LN151@main
push DWORD PTR _s2$[esp+72]
call ??3@YAXPAX@Z           ; operator delete
add esp, 4

```

```
$LN151@main:
    cmp  DWORD PTR _s1$[esp+92], 16
    mov  DWORD PTR _s2$[esp+92], 15
    mov  DWORD PTR _s2$[esp+88], 0
    mov  BYTE PTR _s2$[esp+72], 0
    jb   SHORT $LN195@main
    push DWORD PTR _s1$[esp+72]
    call ??3@YAXPAX@Z           ; operator delete
    add  esp, 4
$LN195@main:
    xor  eax, eax
    add  esp, 72
    ret  0
_main ENDP
```

Listing 51.24: GCC 4.8.1

```
.LC0:
    .string "Hello, "
.LC1:
    .string "world!\n"
main:
    push ebp
    mov  ebp, esp
    push edi
    push esi
    push ebx
    and  esp, -16
    sub  esp, 32
    lea  ebx, [esp+28]
    lea  edi, [esp+20]
    mov  DWORD PTR [esp+8], ebx
    lea  esi, [esp+24]
    mov  DWORD PTR [esp+4], OFFSET FLAT:.LC0
    mov  DWORD PTR [esp], edi

    call _ZNSSc1EPKcRKSaIcE
```

```
    mov  DWORD PTR [esp+8], ebx
    mov  DWORD PTR [esp+4], OFFSET FLAT:.LC1
    mov  DWORD PTR [esp], esi

    call _ZNSsC1EPKcRKSaIcE

    mov  DWORD PTR [esp+4], edi
    mov  DWORD PTR [esp], ebx

    call _ZNSsC1ERKSs

    mov  DWORD PTR [esp+4], esi
    mov  DWORD PTR [esp], ebx

    call _ZNSs6appendERKSs

; c_str():
    mov  eax, DWORD PTR [esp+28]
    mov  DWORD PTR [esp], eax

    call puts

    mov  eax, DWORD PTR [esp+28]
    lea  ebx, [esp+19]
    mov  DWORD PTR [esp+4], ebx
    sub  eax, 12
    mov  DWORD PTR [esp], eax
    call _ZNSs4_Rep10_M_DisposeERKSaIcE
    mov  eax, DWORD PTR [esp+24]
    mov  DWORD PTR [esp+4], ebx
    sub  eax, 12
    mov  DWORD PTR [esp], eax
    call _ZNSs4_Rep10_M_DisposeERKSaIcE
    mov  eax, DWORD PTR [esp+20]
    mov  DWORD PTR [esp+4], ebx
    sub  eax, 12
    mov  DWORD PTR [esp], eax
    call _ZNSs4_Rep10_M_DisposeERKSaIcE
    lea  esp, [ebp-12]
    xor  eax, eax
    pop  ebx
    pop  esi
    pop  edi
    pop  ebp
    ret
```

std::string

```
:
#include <stdio.h>
#include <string>

std::string s="a string";

int main()
{
    printf ("%s\n", s.c_str());
}
```

Listing 51.25: MSVC 2012:

```
?__Es@@YAXXZ PROC
push 8
push OFFSET $SG39512 ; 'a string'
mov  ecx, OFFSET ?s@3V?$basic_string@DU??
    ↴ $char_traits@D@std@V?$allocator@D@2@std@A ; s
call ?assign@?$basic_string@DU?$char_traits@D@std@V??
    ↴ $allocator@D@2@std@QAEAAV12@PBDI@Z ; std::basic_string<,
    ↴ char,std::char_traits<char>,std::allocator<char> >::
    ↴ assign
push OFFSET ??__Fs@@YAXXZ ; `dynamic atexit destructor for '
    ↴ 's'
call _atexit
pop  ecx
ret  0
?__Es@@YAXXZ ENDP
```

Listing 51.26: MSVC 2012: main()

```
$SG39512 DB 'a string', 00H
$SG39519 DB '%s', 0aH, 00H

_main PROC
    cmp  DWORD PTR ?s@3V?$basic_string@DU??
        ↴ $char_traits@D@std@V?$allocator@D@2@std@A+20, 16
    mov  eax, OFFSET ?s@3V?$basic_string@DU??
        ↴ $char_traits@D@std@V?$allocator@D@2@std@A ; s
    cmovae eax, DWORD PTR ?s@3V?$basic_string@DU??
        ↴ $char_traits@D@std@V?$allocator@D@2@std@A
    push eax
    push OFFSET $SG39519 ; '%s'
    call _printf
    add  esp, 8
```

```

    xor  eax, eax
    ret  0
_main ENDP

```

Listing 51.27: MSVC 2012:

```

??_Fs@@YAXXZ PROC
push ecx
cmp  DWORD PTR ?s@@3V?$basic_string@DU?`  

`$char_traits@D@std@V?$allocator@D@2@std@@A+20, 16
jb   SHORT $LN23@dynamic
push esi
mov  esi, DWORD PTR ?s@@3V?$basic_string@DU?`  

`$char_traits@D@std@V?$allocator@D@2@std@@A
lea   ecx, DWORD PTR $T2[esp+8]
call ??0$_Wrap_alloc@V?$allocator@D@std@@std@@QAE@XZ
push OFFSET ?s@@3V?$basic_string@DU?$char_traits@D@std@V?  

`$allocator@D@2@std@@A ; s
lea   ecx, DWORD PTR $T2[esp+12]
call ??$destroy@PAD@$_Wrap_alloc@V?  

`$allocator@D@std@@std@@QAEXPAPAD@Z
lea   ecx, DWORD PTR $T1[esp+8]
call ??0$_Wrap_alloc@V?$allocator@D@std@@std@@QAE@XZ
push esi
call ??3@YAXPAX@Z ; operator delete
add  esp, 4
pop  esi
$LN23@dynamic:
mov  DWORD PTR ?s@@3V?$basic_string@DU?`  

`$char_traits@D@std@V?$allocator@D@2@std@@A+20, 15
mov  DWORD PTR ?s@@3V?$basic_string@DU?`  

`$char_traits@D@std@V?$allocator@D@2@std@@A+16, 0
mov  BYTE PTR ?s@@3V?$basic_string@DU?$char_traits@D@std@V?  

`$allocator@D@2@std@@A, 0
pop  ecx
ret  0
??_Fs@@YAXXZ ENDP

```

.

GCC :

Listing 51.28: GCC 4.8.1

```

main:
    push ebp
    mov  ebp, esp
    and  esp, -16

```

```

    sub esp, 16
    mov eax, DWORD PTR s
    mov DWORD PTR [esp], eax
    call puts
    xor eax, eax
    leave
    ret
.LC0:
    .string "a string"
_GLOBAL__sub_I_s:
    sub esp, 44
    lea eax, [esp+31]
    mov DWORD PTR [esp+8], eax
    mov DWORD PTR [esp+4], OFFSET FLAT:.LC0
    mov DWORD PTR [esp], OFFSET FLAT:s
    call _ZNSsC1EPKcRKSaIcE
    mov DWORD PTR [esp+8], OFFSET FLAT:_dso_handle
    mov DWORD PTR [esp+4], OFFSET FLAT:s
    mov DWORD PTR [esp], OFFSET FLAT:_ZNSsD1Ev
    call __cxa_atexit
    add esp, 44
    ret
.LFE645:
    .size _GLOBAL__sub_I_s, .-_GLOBAL__sub_I_s
    .section .init_array,"aw"
    .align 4
    .long _GLOBAL__sub_I_s
    .globl s
    .bss
    .align 4
    .type s, @object
    .size s, 4
s:
    .zero 4
    .hidden __dso_handle

```

51.4.2 std::list

```
#include <stdio.h>
```

```
#include <list>
#include <iostream>

struct a
{
    int x;
    int y;
};

struct List_node
{
    struct List_node* _Next;
    struct List_node* _Prev;
    int x;
    int y;
};

void dump_List_node (struct List_node *n)
{
    printf ("ptr=0x%p _Next=0x%p _Prev=0x%p x=%d y=%d\n",
            n, n->_Next, n->_Prev, n->x, n->y);
}

void dump_List_vals (struct List_node* n)
{
    struct List_node* current=n;

    for (;;)
    {
        dump_List_node (current);
        current=current->_Next;
        if (current==n) // end
            break;
    };
}

void dump_List_val (unsigned int *a)
{
#ifdef _MSC_VER
    //
    printf ("_Myhead=0x%p, _Mysize=%d\n", a[0], a[1]);
#endif
    dump_List_vals ((struct List_node*)a[0]);
};

int main()
{
```

```
std::list<struct a> l;

printf ("* empty list:\n");
dump_List_val((unsigned int*)(void*)&l);

struct a t1;
t1.x=1;
t1.y=2;
l.push_front (t1);
t1.x=3;
t1.y=4;
l.push_front (t1);
t1.x=5;
t1.y=6;
l.push_back (t1);

printf ("* 3-elements list:\n");
dump_List_val((unsigned int*)(void*)&l);

std::list<struct a>::iterator tmp;
printf ("node at .begin:\n");
tmp=l.begin();
dump_List_node ((struct List_node *)*(void**)&tmp);
printf ("node at .end:\n");
tmp=l.end();
dump_List_node ((struct List_node *)*(void**)&tmp);

printf ("* let's count from the begin:\n");
std::list<struct a>::iterator it=l.begin();
printf ("1st element: %d %d\n", (*it).x, (*it).y);
it++;
printf ("2nd element: %d %d\n", (*it).x, (*it).y);
it++;
printf ("3rd element: %d %d\n", (*it).x, (*it).y);
it++;
printf ("element at .end(): %d %d\n", (*it).x, (*it).y);

printf ("* let's count from the end:\n");
std::list<struct a>::iterator it2=l.end();
printf ("element at .end(): %d %d\n", (*it2).x, (*it2).y);
it2--;
printf ("3rd element: %d %d\n", (*it2).x, (*it2).y);
it2--;
printf ("2nd element: %d %d\n", (*it2).x, (*it2).y);
it2--;
printf ("1st element: %d %d\n", (*it2).x, (*it2).y);
```

```

    printf ("removing last element...\n");
    l.pop_back();
    dump_List_val((unsigned int*)(void*)&l);
};

```

GCC

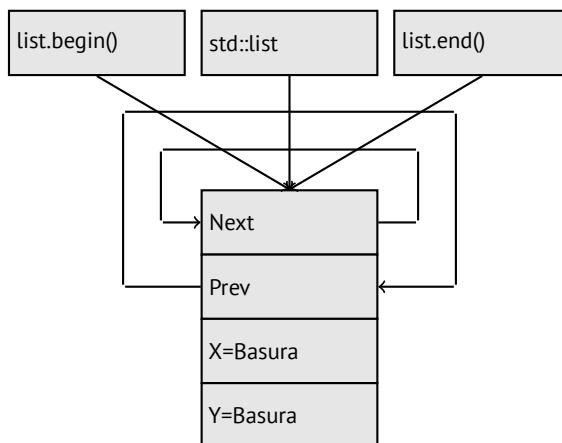
GCC.

```

* empty list:
ptr=0x0028fe90 _Next=0x0028fe90 _Prev=0x0028fe90 x=3 y=0

```

. «next» y «prev» :

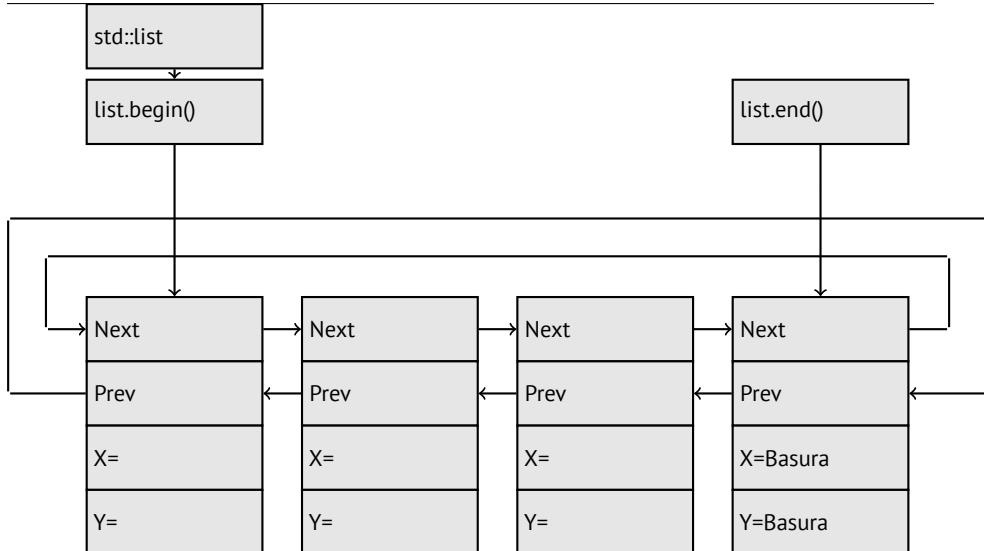


```

* 3-elements list:
ptr=0x000349a0 _Next=0x00034988 _Prev=0x0028fe90 x=3 y=4
ptr=0x00034988 _Next=0x00034b40 _Prev=0x000349a0 x=1 y=2
ptr=0x00034b40 _Next=0x0028fe90 _Prev=0x00034988 x=5 y=6
ptr=0x0028fe90 _Next=0x000349a0 _Prev=0x00034b40 x=5 y=6

```

...
:



l.

list.begin() y list.end() .

```
node at .begin:  
ptr=0x000349a0 _Next=0x00034988 _Prev=0x0028fe90 x=3 y=4  
node at .end:  
ptr=0x0028fe90 _Next=0x000349a0 _Prev=0x00034b40 x=5 y=6
```

*operator-- y operator++ current_node->prev o
current_node->next. (.rbegin, .rend) .*

operator (x).*

dangling pointer.

Listing 51.29: Con optimización GCC 4.8.1 -fno-inline-small-functions

```
main proc near
    push ebp
    mov  ebp, esp
    push esi
```

```

push ebx
and esp, 0FFFFFFF0h
sub esp, 20h
lea ebx, [esp+10h]
mov dword ptr [esp], offset s ; /* empty list:*/
mov [esp+10h], ebx
mov [esp+14h], ebx
call puts
mov [esp], ebx
call _Z13dump_List_valPj ; dump_List_val(uint *)
lea esi, [esp+18h]
mov [esp+4], esi
mov [esp], ebx
mov dword ptr [esp+18h], 1 ; X
mov dword ptr [esp+1Ch], 2 ; Y
call _ZNSt4listI1aSaISO_EE10push_frontERKS0_ ; std::list<a,>::push_front(a const&)
mov [esp+4], esi
mov [esp], ebx
mov dword ptr [esp+18h], 3 ; X
mov dword ptr [esp+1Ch], 4 ; Y
call _ZNSt4listI1aSaISO_EE10push_frontERKS0_ ; std::list<a,>::push_front(a const&)
mov dword ptr [esp], 10h
mov dword ptr [esp+18h], 5 ; X
mov dword ptr [esp+1Ch], 6 ; Y
call _Znwj ; operator new(uint)
cmp eax, 0FFFFFFF8h
jz short loc_80002A6
mov ecx, [esp+1Ch]
mov edx, [esp+18h]
mov [eax+0Ch], ecx
mov [eax+8], edx

loc_80002A6: ; CODE XREF: main+86
mov [esp+4], ebx
mov [esp], eax
call _ZNSt8__detail15_List_node_base7_M_hookEPS0_ ; std::list<__detail::__List_node_base>::M_hook(std::__detail::__List_node_base*)
mov dword ptr [esp], offset a3ElementsList ; /* 3-elements list:*/
call puts
mov [esp], ebx
call _Z13dump_List_valPj ; dump_List_val(uint *)
mov dword ptr [esp], offset aNodeAt_begin ; "node at . begin:"

```

```

call puts
mov  eax, [esp+10h]
mov  [esp], eax
call _Z14dump_List_nodeP9List_node ; dump_List_node(⤦
↳ List_node *)
mov  dword ptr [esp], offset aNodeAt_end ; "node at .end:"
call puts
mov  [esp], ebx
call _Z14dump_List_nodeP9List_node ; dump_List_node(⤦
↳ List_node *)
mov  dword ptr [esp], offset aLetSCountFromT ; "* let's ↤
↳ count from the begin:"
call puts
mov  esi, [esp+10h]
mov  eax, [esi+0Ch]
mov  [esp+0Ch], eax
mov  eax, [esi+8]
mov  dword ptr [esp+4], offset a1stElementDD ; "1st element⤦
↳ : %d\n"
mov  dword ptr [esp], 1
mov  [esp+8], eax
call __printf_chk
mov  esi, [esi] ; operator++: get ->next pointer
mov  eax, [esi+0Ch]
mov  [esp+0Ch], eax
mov  eax, [esi+8]
mov  dword ptr [esp+4], offset a2ndElementDD ; "2nd element⤦
↳ : %d\n"
mov  dword ptr [esp], 1
mov  [esp+8], eax
call __printf_chk
mov  esi, [esi] ; operator++: get ->next pointer
mov  eax, [esi+0Ch]
mov  [esp+0Ch], eax
mov  eax, [esi+8]
mov  dword ptr [esp+4], offset a3rdElementDD ; "3rd element⤦
↳ : %d\n"
mov  dword ptr [esp], 1
mov  [esp+8], eax
call __printf_chk
mov  eax, [esi] ; operator++: get ->next pointer
mov  edx, [eax+0Ch]
mov  [esp+0Ch], edx
mov  eax, [eax+8]
mov  dword ptr [esp+4], offset aElementAt_endD ; "element ↤
↳ at .end(): %d %d\n"
mov  dword ptr [esp], 1

```

```

mov [esp+8], eax
call __printf_chk
mov dword ptr [esp], offset aLetSCountFro_0 ; "* let's ↴
↳ count from the end:"
call puts
mov eax, [esp+1Ch]
mov dword ptr [esp+4], offset aElementAt_endD ; "element ↴
↳ at .end(): %d %d\n"
mov dword ptr [esp], 1
mov [esp+0Ch], eax
mov eax, [esp+18h]
mov [esp+8], eax
call __printf_chk
mov esi, [esp+14h]
mov eax, [esi+0Ch]
mov [esp+0Ch], eax
mov eax, [esi+8]
mov dword ptr [esp+4], offset a3rdElementDD ; "3rd element ↴
↳ : %d %d\n"
mov dword ptr [esp], 1
mov [esp+8], eax
call __printf_chk
mov esi, [esi+4] ; operator--: get ->prev pointer
mov eax, [esi+0Ch]
mov [esp+0Ch], eax
mov eax, [esi+8]
mov dword ptr [esp+4], offset a2ndElementDD ; "2nd element ↴
↳ : %d %d\n"
mov dword ptr [esp], 1
mov [esp+8], eax
call __printf_chk
mov eax, [esi+4] ; operator--: get ->prev pointer
mov edx, [eax+0Ch]
mov [esp+0Ch], edx
mov eax, [eax+8]
mov dword ptr [esp+4], offset a1stElementDD ; "1st element ↴
↳ : %d %d\n"
mov dword ptr [esp], 1
mov [esp+8], eax
call __printf_chk
mov dword ptr [esp], offset aRemovingLastEl ; "removing ↴
↳ last element..."
call puts
mov esi, [esp+14h]
mov [esp], esi
call __ZNSt8__detail15_List_node_base9_M_unhookEv ; std:: ↴
↳ __detail::__List_node_base::__M_unhook(void)

```

```

    mov [esp], esi ; void *
    call _ZdlPv ; operator delete(void *)
    mov [esp], ebx
    call _Z13dump_List_valPj ; dump_List_val(uint *)
    mov [esp], ebx
    call _ZNSt10_List_baseI1aSaIS0_EE8_M_clearEv ; std::list<a>::_M_clear(void)
    ↴ _List_base<a, std::allocator<a>>::_M_clear(void)
    lea esp, [ebp-8]
    xor eax, eax
    pop ebx
    pop esi
    pop ebp
    retn
main endp

```

Listing 51.30:

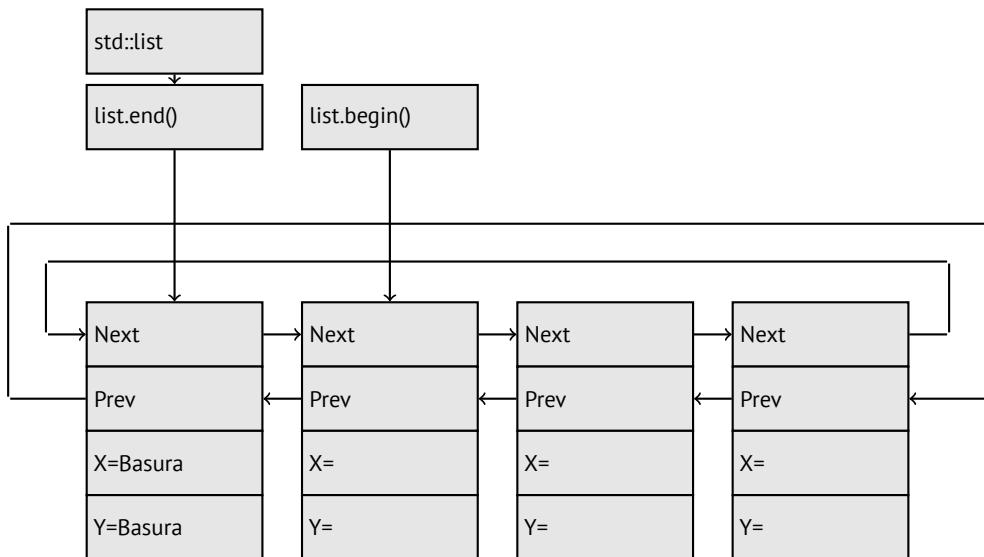
```

* empty list:
ptr=0x0028fe90 _Next=0x0028fe90 _Prev=0x0028fe90 x=3 y=0
* 3-elements list:
ptr=0x000349a0 _Next=0x00034988 _Prev=0x0028fe90 x=3 y=4
ptr=0x00034988 _Next=0x00034b40 _Prev=0x000349a0 x=1 y=2
ptr=0x00034b40 _Next=0x0028fe90 _Prev=0x00034988 x=5 y=6
ptr=0x0028fe90 _Next=0x000349a0 _Prev=0x00034b40 x=5 y=6
node at .begin:
ptr=0x000349a0 _Next=0x00034988 _Prev=0x0028fe90 x=3 y=4
node at .end:
ptr=0x0028fe90 _Next=0x000349a0 _Prev=0x00034b40 x=5 y=6
* let's count from the begin:
1st element: 3 4
2nd element: 1 2
3rd element: 5 6
element at .end(): 5 6
* let's count from the end:
element at .end(): 5 6
3rd element: 5 6
2nd element: 1 2
1st element: 3 4
removing last element...
ptr=0x000349a0 _Next=0x00034988 _Prev=0x0028fe90 x=3 y=4
ptr=0x00034988 _Next=0x0028fe90 _Prev=0x000349a0 x=1 y=2
ptr=0x0028fe90 _Next=0x000349a0 _Prev=0x00034988 x=5 y=6

```

MSVC

. . .



Listing 51.31: Con optimización MSVC 2012 /Fa2.asm /GS- /Ob1

```

_l$ = -16 ; size = 8
_t1$ = -8 ; size = 8
_main PROC
    sub esp, 16
    push ebx
    push esi
    push edi
    push 0
    push 0
    lea ecx, DWORD PTR _l$[esp+36]
    mov DWORD PTR _l$[esp+40], 0
    ;
    call ?_Buynode0@?$_List_alloc@$0A@U?$_List_base_types@Ua@@V
    ↴ ?$allocator@Ua@@@std@@@std@@@std@@QAEPAU? ↴
    ↴ _$List_node@Ua@@PAX@2@PAU32@0@Z ; std::_List_alloc<0,std
    ↴ :::_List_base_types<a,std::allocator<a> > >:_Buynode0
    mov edi, DWORD PTR __imp_printf
    mov ebx, eax
    push OFFSET $SG40685 ; '* empty list:'
    mov DWORD PTR _l$[esp+32], ebx
    call edi ; printf
    lea eax, DWORD PTR _l$[esp+32]
    push eax
    call ?dump_List_val@@YAXPAI@Z ; dump_List_val

```

```

    mov  esi, DWORD PTR [ebx]
    add  esp, 8
    lea  eax, DWORD PTR _t1$[esp+28]
    push eax
    push DWORD PTR [esi+4]
    lea  ecx, DWORD PTR _l$[esp+36]
    push esi
    mov  DWORD PTR _t1$[esp+40], 1 ;
    mov  DWORD PTR _t1$[esp+44], 2 ;
    ; allocate new node
    call ??$_Buynode@ABUa@@@?$_List_buy@Ua@@V?↵
    ↴ $allocator@Ua@@@std@@@std@@QAEPAU?↵
    ↴ $_List_node@Ua@@PAX@1@PAU21@0ABUa@@@Z ; std::_List_buy<a,>
    ↴ std::allocator<a> >::_Buynode<a const &>
    mov  DWORD PTR [esi+4], eax
    mov  ecx, DWORD PTR [eax+4]
    mov  DWORD PTR _t1$[esp+28], 3 ;
    mov  DWORD PTR [ecx], eax
    mov  esi, DWORD PTR [ebx]
    lea  eax, DWORD PTR _t1$[esp+28]
    push eax
    push DWORD PTR [esi+4]
    lea  ecx, DWORD PTR _l$[esp+36]
    push esi
    mov  DWORD PTR _t1$[esp+44], 4 ;
    ; allocate new node
    call ??$_Buynode@ABUa@@@?$_List_buy@Ua@@V?↵
    ↴ $allocator@Ua@@@std@@@std@@QAEPAU?↵
    ↴ $_List_node@Ua@@PAX@1@PAU21@0ABUa@@@Z ; std::_List_buy<a,>
    ↴ std::allocator<a> >::_Buynode<a const &>
    mov  DWORD PTR [esi+4], eax
    mov  ecx, DWORD PTR [eax+4]
    mov  DWORD PTR _t1$[esp+28], 5 ;
    mov  DWORD PTR [ecx], eax
    lea  eax, DWORD PTR _t1$[esp+28]
    push eax
    push DWORD PTR [ebx+4]
    lea  ecx, DWORD PTR _l$[esp+36]
    push ebx
    mov  DWORD PTR _t1$[esp+44], 6 ;
    ; allocate new node
    call ??$_Buynode@ABUa@@@?$_List_buy@Ua@@V?↵
    ↴ $allocator@Ua@@@std@@@std@@QAEPAU?↵
    ↴ $_List_node@Ua@@PAX@1@PAU21@0ABUa@@@Z ; std::_List_buy<a,>
    ↴ std::allocator<a> >::_Buynode<a const &>
    mov  DWORD PTR [ebx+4], eax
    mov  ecx, DWORD PTR [eax+4]

```

```

push OFFSET $SG40689 ; '* 3-elements list:'
mov DWORD PTR _l$[esp+36], 3
mov DWORD PTR [ecx], eax
call edi ; printf
lea eax, DWORD PTR _l$[esp+32]
push eax
call ?dump_List_val@@YAXPAI@Z ; dump_List_val
push OFFSET $SG40831 ; 'node at .begin:'
call edi ; printf
push DWORD PTR [ebx] ;
call ?dump_List_node@@YAXPAUList_node@@@Z ; dump_List_node
push OFFSET $SG40835 ; 'node at .end:'
call edi ; printf
push ebx ; pointer to the node $l$ variable points to!
call ?dump_List_node@@YAXPAUList_node@@@Z ; dump_List_node
push OFFSET $SG40839 ; '* let''s count from the begin:'
call edi ; printf
mov esi, DWORD PTR [ebx] ; operator++: get ->next pointer
push DWORD PTR [esi+12]
push DWORD PTR [esi+8]
push OFFSET $SG40846 ; '1st element: %d %d'
call edi ; printf
mov esi, DWORD PTR [esi] ; operator++: get ->next pointer
push DWORD PTR [esi+12]
push DWORD PTR [esi+8]
push OFFSET $SG40848 ; '2nd element: %d %d'
call edi ; printf
mov esi, DWORD PTR [esi] ; operator++: get ->next pointer
push DWORD PTR [esi+12]
push DWORD PTR [esi+8]
push OFFSET $SG40850 ; '3rd element: %d %d'
call edi ; printf
mov eax, DWORD PTR [esi] ; operator++: get ->next pointer
add esp, 64
push DWORD PTR [eax+12]
push DWORD PTR [eax+8]
push OFFSET $SG40852 ; 'element at .end(): %d %d'
call edi ; printf
push OFFSET $SG40853 ; '* let''s count from the end:'
call edi ; printf
push DWORD PTR [ebx+12] ;
push DWORD PTR [ebx+8]
push OFFSET $SG40860 ; 'element at .end(): %d %d'
call edi ; printf
mov esi, DWORD PTR [ebx+4] ; operator--: get ->prev ↲
pointer
push DWORD PTR [esi+12]

```

```

push DWORD PTR [esi+8]
push OFFSET $SG40862 ; '3rd element: %d %d'
call edi ; printf
mov esi, DWORD PTR [esi+4] ; operator--: get ->prev ↵
↳ pointer
push DWORD PTR [esi+12]
push DWORD PTR [esi+8]
push OFFSET $SG40864 ; '2nd element: %d %d'
call edi ; printf
mov eax, DWORD PTR [esi+4] ; operator--: get ->prev ↵
↳ pointer
push DWORD PTR [eax+12]
push DWORD PTR [eax+8]
push OFFSET $SG40866 ; '1st element: %d %d'
call edi ; printf
add esp, 64
push OFFSET $SG40867 ; 'removing last element...'
call edi ; printf
mov edx, DWORD PTR [ebx+4]
add esp, 4

; prev=next?
; ?
; !
cmp edx, ebx
je SHORT $LN349@main
mov ecx, DWORD PTR [edx+4]
mov eax, DWORD PTR [edx]
mov DWORD PTR [ecx], eax
mov ecx, DWORD PTR [edx]
mov eax, DWORD PTR [edx+4]
push edx
mov DWORD PTR [ecx+4], eax
call ??3@YAXPAX@Z ; operator delete
add esp, 4
mov DWORD PTR _l$[esp+32], 2
$LN349@main:
lea eax, DWORD PTR _l$[esp+28]
push eax
call ?dump_List_val@@YAXPAI@Z ; dump_List_val
mov eax, DWORD PTR [ebx]
add esp, 4
mov DWORD PTR [ebx], ebx
mov DWORD PTR [ebx+4], ebx
cmp eax, ebx
je SHORT $LN412@main
$LL414@main:

```

```

    mov    esi, DWORD PTR [eax]
    push   eax
    call   ??3@YAXPAX@Z ; operator delete
    add    esp, 4
    mov    eax, esi
    cmp    esi, ebx
    jne    SHORT $LL414@main
$LN412@main:
    push   ebx
    call   ??3@YAXPAX@Z ; operator delete
    add    esp, 4
    xor    eax, eax
    pop    edi
    pop    esi
    pop    ebx
    add    esp, 16
    ret    0
_main ENDP

```

().

Listing 51.32:

```

* empty list:
_Myhead=0x003CC258, _Mysize=0
ptr=0x003CC258 _Next=0x003CC258 _Prev=0x003CC258 x=6226002 y=4
    ↴ =4522072
* 3-elements list:
_Myhead=0x003CC258, _Mysize=3
ptr=0x003CC258 _Next=0x003CC288 _Prev=0x003CC2A0 x=6226002 y=4
    ↴ =4522072
ptr=0x003CC288 _Next=0x003CC270 _Prev=0x003CC258 x=3 y=4
ptr=0x003CC270 _Next=0x003CC2A0 _Prev=0x003CC288 x=1 y=2
ptr=0x003CC2A0 _Next=0x003CC258 _Prev=0x003CC270 x=5 y=6
node at .begin:
ptr=0x003CC288 _Next=0x003CC270 _Prev=0x003CC258 x=3 y=4
node at .end:
ptr=0x003CC258 _Next=0x003CC288 _Prev=0x003CC2A0 x=6226002 y=4
    ↴ =4522072
* let's count from the begin:
1st element: 3 4
2nd element: 1 2
3rd element: 5 6
element at .end(): 6226002 4522072
* let's count from the end:
element at .end(): 6226002 4522072
3rd element: 5 6
2nd element: 1 2

```

```
1st element: 3 4
removing last element...
_Myhead=0x003CC258, _Mysize=2
ptr=0x003CC258 _Next=0x003CC288 _Prev=0x003CC270 x=6226002 y=4
    ↴ =4522072
ptr=0x003CC288 _Next=0x003CC270 _Prev=0x003CC258 x=3 y=4
ptr=0x003CC270 _Next=0x003CC258 _Prev=0x003CC288 x=1 y=2
```

C++11 std::forward_list

..

51.4.3 std::vector

std::string (51.4.1 on page 723):

(18 on page 326). C++11 .data() .c_str() en std::string.
³, std::vector:

```
#include <stdio.h>
#include <vector>
#include <algorithm>
#include <functional>

struct vector_of_ints
{
    // MSVC names:
    int *Myfirst;
    int *Mylast;
    int *Myend;

    // : _M_start, _M_finish, _M_end_of_storage
};

void dump(struct vector_of_ints *in)
{
    printf ("_Myfirst=%p, _Mylast=%p, _Myend=%p\n", in->Myfirst,
    ↴ , in->Mylast, in->Myend);
    size_t size=(in->Mylast-in->Myfirst);
    size_t capacity=(in->Myend-in->Myfirst);
    printf ("size=%d, capacity=%d\n", size, capacity);
    for (size_t i=0; i<size; i++)
        printf ("element %d: %d\n", i, in->Myfirst[i]);
```

³: <http://go.yurichev.com/17086>

```

};

int main()
{
    std::vector<int> c;
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(1);
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(2);
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(3);
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(4);
    dump ((struct vector_of_ints*)(void*)&c);
    c.reserve (6);
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(5);
    dump ((struct vector_of_ints*)(void*)&c);
    c.push_back(6);
    dump ((struct vector_of_ints*)(void*)&c);
    printf ("%d\n", c.at(5)); //
    printf ("%d\n", c[8]); // operator[],
};

```

MSVC:

```

_Myfirst=00000000, _Mylast=00000000, _Myend=00000000
size=0, capacity=0
_Myfirst=0051CF48, _Mylast=0051CF4C, _Myend=0051CF4C
size=1, capacity=1
element 0: 1
_Myfirst=0051CF58, _Mylast=0051CF60, _Myend=0051CF60
size=2, capacity=2
element 0: 1
element 1: 2
_Myfirst=0051C278, _Mylast=0051C284, _Myend=0051C284
size=3, capacity=3
element 0: 1
element 1: 2
element 2: 3
_Myfirst=0051C290, _Mylast=0051C2A0, _Myend=0051C2A0
size=4, capacity=4
element 0: 1
element 1: 2
element 2: 3
element 3: 4
_Myfirst=0051B180, _Mylast=0051B190, _Myend=0051B198
size=4, capacity=6

```

```

element 0: 1
element 1: 2
element 2: 3
element 3: 4
_Myfirst=0051B180, _Mylast=0051B194, _Myend=0051B198
size=5, capacity=6
element 0: 1
element 1: 2
element 2: 3
element 3: 4
element 4: 5
_Myfirst=0051B180, _Mylast=0051B198, _Myend=0051B198
size=6, capacity=6
element 0: 1
element 1: 2
element 2: 3
element 3: 4
element 4: 5
element 5: 6
6
6619158

```

. push_back(). push_back() . push_back() .. reserve().. operator[]
 std::vector . .at() std::out_of_range.

:

Listing 51.33: MSVC 2012 /GS- /Ob1

```

$SG52650 DB '%d', 0aH, 00H
$SG52651 DB '%d', 0aH, 00H

_this$ = -4 ; size = 4
__Pos$ = 8 ; size = 4
?at@?$vector@HV?$allocator@H@std@@@std@@QAEAAHI@Z PROC ; std::vector<int, std::allocator<int> >::at, COMDAT
    ↴ vector<int, std::allocator<int> >::at, COMDAT
; _this$ = ecx
    push ebp
    mov ebp, esp
    push ecx
    mov DWORD PTR _this$[ebp], ecx
    mov eax, DWORD PTR _this$[ebp]
    mov ecx, DWORD PTR _this$[ebp]
    mov edx, DWORD PTR [eax+4]
    sub edx, DWORD PTR [ecx]
    sar edx, 2
    cmp edx, DWORD PTR __Pos$[ebp]
    ja SHORT $LN1@at

```

```

push OFFSET ??_C@_0BM@NMJKDPP0@invalid?5vector?$DMT?$DO?5@
↳ subscript?$AA@
call DWORD PTR __imp_?Xout_of_range@std@@YAXPBD@Z
$LN1@at:
    mov    eax, DWORD PTR _this$[ebp]
    mov    ecx, DWORD PTR [eax]
    mov    edx, DWORD PTR __Pos$[ebp]
    lea    eax, DWORD PTR [ecx+edx*4]
$LN3@at:
    mov    esp, ebp
    pop    ebp
    ret    4
?at@?$vector@HV?$allocator@H@std@@@std@@QAEAAHI@Z ENDP ; std::vector<int, std::allocator<int> >::at

_c$ = -36 ; size = 12
$T1 = -24 ; size = 4
$T2 = -20 ; size = 4
$T3 = -16 ; size = 4
$T4 = -12 ; size = 4
$T5 = -8  ; size = 4
$T6 = -4  ; size = 4
_main PROC
    push  ebp
    mov   ebp, esp
    sub   esp, 36
    mov   DWORD PTR _c$[ebp], 0      ; Myfirst
    mov   DWORD PTR _c$[ebp+4], 0    ; Mylast
    mov   DWORD PTR _c$[ebp+8], 0    ; Myend
    lea   eax, DWORD PTR _c$[ebp]
    push  eax
    call ?dump@YAXPAUvector_of_ints@@@Z ; dump
    add   esp, 4
    mov   DWORD PTR $T6[ebp], 1
    lea   ecx, DWORD PTR $T6[ebp]
    push  ecx
    lea   ecx, DWORD PTR _c$[ebp]
    call ?push_back@?$vector@HV?@
↳ $allocator@H@std@@@std@@QAE$QAH@Z ; std::vector<int, std::allocator<int> >::push_back
    lea   edx, DWORD PTR _c$[ebp]
    push  edx
    call ?dump@YAXPAUvector_of_ints@@@Z ; dump
    add   esp, 4
    mov   DWORD PTR $T5[ebp], 2
    lea   eax, DWORD PTR $T5[ebp]
    push  eax

```

```
lea  ecx, DWORD PTR _c$[ebp]
call ?push_back@?$vector@HV?>
↳ $allocator@H@std@@@std@@@QEXX$QAH@Z ; std::vector<int, stdx
↳ ::allocator<int> >::push_back
lea  ecx, DWORD PTR _c$[ebp]
push ecx
call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add  esp, 4
mov  DWORD PTR $T4[ebp], 3
lea  edx, DWORD PTR $T4[ebp]
push edx
lea  ecx, DWORD PTR _c$[ebp]
call ?push_back@?$vector@HV?>
↳ $allocator@H@std@@@std@@@QEXX$QAH@Z ; std::vector<int, stdx
↳ ::allocator<int> >::push_back
lea  eax, DWORD PTR _c$[ebp]
push eax
call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add  esp, 4
mov  DWORD PTR $T3[ebp], 4
lea  ecx, DWORD PTR $T3[ebp]
push ecx
lea  ecx, DWORD PTR _c$[ebp]
call ?push_back@?$vector@HV?>
↳ $allocator@H@std@@@std@@@QEXX$QAH@Z ; std::vector<int, stdx
↳ ::allocator<int> >::push_back
lea  edx, DWORD PTR _c$[ebp]
push edx
call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add  esp, 4
push 6
lea  ecx, DWORD PTR _c$[ebp]
call ?reserve@?$vector@HV?$allocator@H@std@@@std@@@QEXI@Z ; std
↳ ::vector<int, std::allocator<int>>::reserve
lea  eax, DWORD PTR _c$[ebp]
push eax
call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add  esp, 4
mov  DWORD PTR $T2[ebp], 5
lea  ecx, DWORD PTR $T2[ebp]
push ecx
lea  ecx, DWORD PTR _c$[ebp]
call ?push_back@?$vector@HV?>
↳ $allocator@H@std@@@std@@@QEXX$QAH@Z ; std::vector<int, stdx
↳ ::allocator<int> >::push_back
lea  edx, DWORD PTR _c$[ebp]
push edx
```

```

call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add esp, 4
mov DWORD PTR $T1[ebp], 6
lea eax, DWORD PTR $T1[ebp]
push eax
lea ecx, DWORD PTR _c$[ebp]
call ?push_back@?$vector@HV?@$allocator@H@std@@@std@@QAEAX$QAH@Z ; std::vector<int, std::allocator<int> >::push_back
lea ecx, DWORD PTR _c$[ebp]
push ecx
call ?dump@@YAXPAUvector_of_ints@@@Z ; dump
add esp, 4
push 5
lea ecx, DWORD PTR _c$[ebp]
call ?at@?$vector@HV?@$allocator@H@std@@@std@@QAEAAHI@Z ; std::vector<int, std::allocator<int> >::at
mov edx, DWORD PTR [eax]
push edx
push OFFSET $SG52650 ; '%d'
call DWORD PTR __imp__printf
add esp, 8
mov eax, 8
shl eax, 2
mov ecx, DWORD PTR _c$[ebp]
mov edx, DWORD PTR [ecx+eax]
push edx
push OFFSET $SG52651 ; '%d'
call DWORD PTR __imp__printf
add esp, 8
lea ecx, DWORD PTR _c$[ebp]
call ?_Tidy@?$vector@HV?@$allocator@H@std@@@std@@IAEXXZ ; std::vector<int, std::allocator<int> >::_Tidy
xor eax, eax
mov esp, ebp
pop ebp
ret 0
_main ENDP

```

.at() .printf().
«size» y «capacity», std::string..
.at():

Listing 51.34: GCC 4.8.1 -fno-inline-small-functions -O1

```

main proc near
    push ebp

```

```
    mov  ebp, esp
    push edi
    push esi
    push ebx
    and  esp, 0FFFFFFF0h
    sub  esp, 20h
    mov  dword ptr [esp+14h], 0
    mov  dword ptr [esp+18h], 0
    mov  dword ptr [esp+1Ch], 0
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov  dword ptr [esp+10h], 1
    lea  eax, [esp+10h]
    mov  [esp+4], eax
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,>
    ↴ std::allocator<int>>::push_back(int const&)
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov  dword ptr [esp+10h], 2
    lea  eax, [esp+10h]
    mov  [esp+4], eax
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,>
    ↴ std::allocator<int>>::push_back(int const&)
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov  dword ptr [esp+10h], 3
    lea  eax, [esp+10h]
    mov  [esp+4], eax
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,>
    ↴ std::allocator<int>>::push_back(int const&)
    lea  eax, [esp+14h]
    mov  [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov  dword ptr [esp+10h], 4
    lea  eax, [esp+10h]
    mov  [esp+4], eax
    lea  eax, [esp+14h]
    mov  [esp], eax
```

```
call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,>
    ↳ std::allocator<int>::push_back(int const&)
    lea eax, [esp+14h]
    mov [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov ebx, [esp+14h]
    mov eax, [esp+1Ch]
    sub eax, ebx
    cmp eax, 17h
    ja short loc_80001CF
    mov edi, [esp+18h]
    sub edi, ebx
    sar edi, 2
    mov dword ptr [esp], 18h
    call _Znwj           ; operator new(uint)
    mov esi, eax
    test edi, edi
    jz short loc_80001AD
    lea eax, ds:0[edi*4]
    mov [esp+8], eax     ; n
    mov [esp+4], ebx     ; src
    mov [esp], esi       ; dest
    call memmove

loc_80001AD: ; CODE XREF: main+F8
    mov eax, [esp+14h]
    test eax, eax
    jz short loc_80001BD
    mov [esp], eax      ; void *
    call _ZdlPv          ; operator delete(void *)

loc_80001BD: ; CODE XREF: main+117
    mov [esp+14h], esi
    lea eax, [esi+edi*4]
    mov [esp+18h], eax
    add esi, 18h
    mov [esp+1Ch], esi

loc_80001CF: ; CODE XREF: main+DD
    lea eax, [esp+14h]
    mov [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov dword ptr [esp+10h], 5
    lea eax, [esp+10h]
    mov [esp+4], eax
    lea eax, [esp+14h]
    mov [esp], eax
```

```

call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,⤦
↳ std::allocator<int>>::push_back(int const&)
    lea    eax, [esp+14h]
    mov    [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov    dword ptr [esp+10h], 6
    lea    eax, [esp+10h]
    mov    [esp+4], eax
    lea    eax, [esp+14h]
    mov    [esp], eax
    call _ZNSt6vectorIiSaIiEE9push_backERKi ; std::vector<int,⤦
↳ std::allocator<int>>::push_back(int const&)
    lea    eax, [esp+14h]
    mov    [esp], eax
    call _Z4dumpP14vector_of_ints ; dump(vector_of_ints *)
    mov    eax, [esp+14h]
    mov    edx, [esp+18h]
    sub    edx, eax
    cmp    edx, 17h
    ja    short loc_8000246
    mov    dword ptr [esp], offset aVector_m_range ; "vector::⤦
↳ _M_range_check"
    call _ZSt20__throw_out_of_rangePKc ; std::⤦
↳ __throw_out_of_range(char const*)
loc_8000246:                                ; CODE XREF: main+19C
    mov    eax, [eax+14h]
    mov    [esp+8], eax
    mov    dword ptr [esp+4], offset aD ; "%d\n"
    mov    dword ptr [esp], 1
    call __printf_chk
    mov    eax, [esp+14h]
    mov    eax, [eax+20h]
    mov    [esp+8], eax
    mov    dword ptr [esp+4], offset aD ; "%d\n"
    mov    dword ptr [esp], 1
    call __printf_chk
    mov    eax, [esp+14h]
    test   eax, eax
    jz    short loc_80002AC
    mov    [esp], eax      ; void *
    call _ZdlPv           ; operator delete(void *)
    jmp    short loc_80002AC

    mov    ebx, eax
    mov    edx, [esp+14h]
    test   edx, edx

```

```

jz    short loc_80002A4
mov  [esp], edx      ; void *
call _ZdlPv          ; operator delete(void *)

loc_80002A4: ; CODE XREF: main+1FE
    mov  [esp], ebx
    call _Unwind_Resume

loc_80002AC: ; CODE XREF: main+1EA
    ; main+1F4
    mov  eax, 0
    lea  esp, [ebp-0Ch]
    pop  ebx
    pop  esi
    pop  edi
    pop  ebp

locret_80002B8: ; DATA XREF: .eh_frame:08000510
    ; .eh_frame:080005BC
    retn
main endp

```

.reserve(). new() , memmove(), delete() .

:

```

_Myfirst=0x(nil), _Mylast=0x(nil), _Myend=0x(nil)
size=0, capacity=0
_Myfirst=0x8257008, _Mylast=0x825700c, _Myend=0x825700c
size=1, capacity=1
element 0: 1
_Myfirst=0x8257018, _Mylast=0x8257020, _Myend=0x8257020
size=2, capacity=2
element 0: 1
element 1: 2
_Myfirst=0x8257028, _Mylast=0x8257034, _Myend=0x8257038
size=3, capacity=4
element 0: 1
element 1: 2
element 2: 3
_Myfirst=0x8257028, _Mylast=0x8257038, _Myend=0x8257038
size=4, capacity=4
element 0: 1
element 1: 2
element 2: 3
element 3: 4
_Myfirst=0x8257040, _Mylast=0x8257050, _Myend=0x8257058

```

```
size=4, capacity=6
element 0: 1
element 1: 2
element 2: 3
element 3: 4
_Myfirst=0x8257040, _Mylast=0x8257054, _Myend=0x8257058
size=5, capacity=6
element 0: 1
element 1: 2
element 2: 3
element 3: 4
element 4: 5
_Myfirst=0x8257040, _Mylast=0x8257058, _Myend=0x8257058
size=6, capacity=6
element 0: 1
element 1: 2
element 2: 3
element 3: 4
element 4: 5
element 5: 6
6
0
```

MSVC.

~50%, 100% .

51.4.4 std::map y std::set

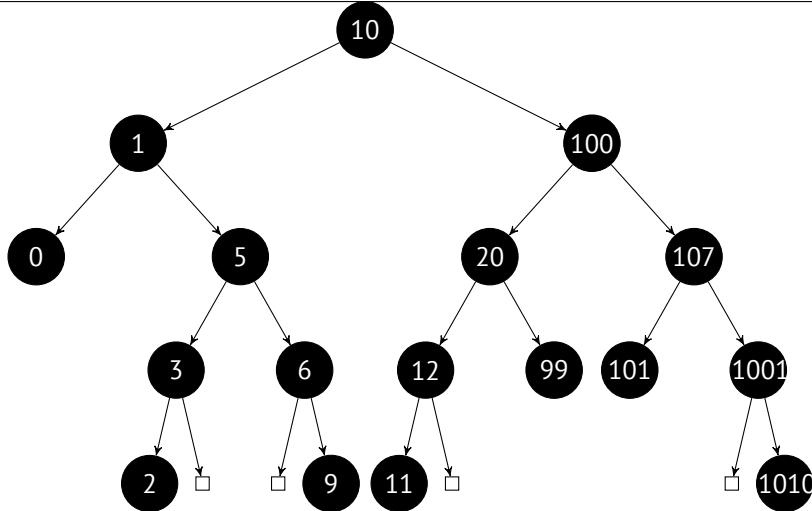
...

.

• ..

.

: 0, 1, 2, 3, 5, 6, 9, 10, 11, 12, 20, 99, 100, 101, 107, 1001, 1010.



$\approx \log_2 n . \approx 10 \approx 1000 \approx 13 \approx 10000 \dots$

std::set std::map .

MSVC

```

#include <map>
#include <set>
#include <string>
#include <iostream>

// !
struct tree_node
{
    struct tree_node *Left;
    struct tree_node *Parent;
    struct tree_node *Right;
    char Color; // 0 - Red, 1 - Black
    char Isnil;
    //std::pair Myval;
  
```



```

        printf ("%d [%s]\n", n->first, n->second);
    if (n->Left->Isnil==0)
    {
        printf ("%.*sL-----", tabs, ALOT_OF_TABS);
        dump_as_tree (tabs+1, n->Left, is_set);
    };
    if (n->Right->Isnil==0)
    {
        printf ("%.*sR-----", tabs, ALOT_OF_TABS);
        dump_as_tree (tabs+1, n->Right, is_set);
    };
};

void dump_map_and_set(struct tree_struct *m, bool is_set)
{
    printf ("ptr=0x%p, Myhead=0x%p, Mysize=%d\n", m, m->Myhead, ↴
    ↴ m->Mysize);
    dump_tree_node (m->Myhead, is_set, true);
    printf ("As a tree:\n");
    printf ("root---");
    dump_as_tree (1, m->Myhead->Parent, is_set);
};

int main()
{
    // map

    std::map<int, const char*> m;

    m[10]="ten";
    m[20]="twenty";
    m[3]="three";
    m[101]="one hundred one";
    m[100]="one hundred";
    m[12]="twelve";
    m[107]="one hundred seven";
    m[0]="zero";
    m[1]="one";
    m[6]="six";
    m[99]="ninety-nine";
    m[5]="five";
    m[11]="eleven";
    m[1001]="one thousand one";
    m[1010]="one thousand ten";
    m[2]="two";
    m[9]="nine";
    printf ("dumping m as map:\n");
}

```

```

dump_map_and_set ((struct tree_struct *)(void*)&m, false);

std::map<int, const char*>::iterator it1=m.begin();
printf ("m.begin():\n");
dump_tree_node ((struct tree_node *)*(void**)&it1, false, ↴
↳ false);
it1=m.end();
printf ("m.end():\n");
dump_tree_node ((struct tree_node *)*(void**)&it1, false, ↴
↳ false);

// set

std::set<int> s;
s.insert(123);
s.insert(456);
s.insert(11);
s.insert(12);
s.insert(100);
s.insert(1001);
printf ("dumping s as set:\n");
dump_map_and_set ((struct tree_struct *)(void*)&s, true);
std::set<int>::iterator it2=s.begin();
printf ("s.begin():\n");
dump_tree_node ((struct tree_node *)*(void**)&it2, true, ↴
↳ false);
it2=s.end();
printf ("s.end():\n");
dump_tree_node ((struct tree_node *)*(void**)&it2, true, ↴
↳ false);
};

}

```

Listing 51.35: MSVC 2012

```

dumping m as map:
ptr=0x0020FE04, Myhead=0x005BB3A0, Mysize=17
ptr=0x005BB3A0 Left=0x005BB4A0 Parent=0x005BB3C0 Right=0 ↴
↳ x005BB580 Color=1 Isnil=1
ptr=0x005BB3C0 Left=0x005BB4C0 Parent=0x005BB3A0 Right=0 ↴
↳ x005BB440 Color=1 Isnil=0
first=10 second=[ten]
ptr=0x005BB4C0 Left=0x005BB4A0 Parent=0x005BB3C0 Right=0 ↴
↳ x005BB520 Color=1 Isnil=0
first=1 second=[one]
ptr=0x005BB4A0 Left=0x005BB3A0 Parent=0x005BB4C0 Right=0 ↴
↳ x005BB3A0 Color=1 Isnil=0
first=0 second=[zero]

```

```

ptr=0x005BB520 Left=0x005BB400 Parent=0x005BB4C0 Right=0✓
    ↴ x005BB4E0 Color=0 Isnil=0
first=5 second=[five]
ptr=0x005BB400 Left=0x005BB5A0 Parent=0x005BB520 Right=0✓
    ↴ x005BB3A0 Color=1 Isnil=0
first=3 second=[three]
ptr=0x005BB5A0 Left=0x005BB3A0 Parent=0x005BB400 Right=0✓
    ↴ x005BB3A0 Color=0 Isnil=0
first=2 second=[two]
ptr=0x005BB4E0 Left=0x005BB3A0 Parent=0x005BB520 Right=0✓
    ↴ x005BB5C0 Color=1 Isnil=0
first=6 second=[six]
ptr=0x005BB5C0 Left=0x005BB3A0 Parent=0x005BB4E0 Right=0✓
    ↴ x005BB3A0 Color=0 Isnil=0
first=9 second=[nine]
ptr=0x005BB440 Left=0x005BB3E0 Parent=0x005BB3C0 Right=0✓
    ↴ x005BB480 Color=1 Isnil=0
first=100 second=[one hundred]
ptr=0x005BB3E0 Left=0x005BB460 Parent=0x005BB440 Right=0✓
    ↴ x005BB500 Color=0 Isnil=0
first=20 second=[twenty]
ptr=0x005BB460 Left=0x005BB540 Parent=0x005BB3E0 Right=0✓
    ↴ x005BB3A0 Color=1 Isnil=0
first=12 second=[twelve]
ptr=0x005BB540 Left=0x005BB3A0 Parent=0x005BB460 Right=0✓
    ↴ x005BB3A0 Color=0 Isnil=0
first=11 second=[eleven]
ptr=0x005BB500 Left=0x005BB3A0 Parent=0x005BB3E0 Right=0✓
    ↴ x005BB3A0 Color=1 Isnil=0
first=99 second=[ninety-nine]
ptr=0x005BB480 Left=0x005BB420 Parent=0x005BB440 Right=0✓
    ↴ x005BB560 Color=0 Isnil=0
first=107 second=[one hundred seven]
ptr=0x005BB420 Left=0x005BB3A0 Parent=0x005BB480 Right=0✓
    ↴ x005BB3A0 Color=1 Isnil=0
first=101 second=[one hundred one]
ptr=0x005BB560 Left=0x005BB3A0 Parent=0x005BB480 Right=0✓
    ↴ x005BB580 Color=1 Isnil=0
first=1001 second=[one thousand one]
ptr=0x005BB580 Left=0x005BB3A0 Parent=0x005BB560 Right=0✓
    ↴ x005BB3A0 Color=0 Isnil=0
first=1010 second=[one thousand ten]
As a tree:
root----10 [ten]
    L-----1 [one]
        L-----0 [zero]
        R-----5 [five]

```

```

L-----3 [three]
    L-----2 [two]
    R-----6 [six]
        R-----9 [nine]
R-----100 [one hundred]
    L-----20 [twenty]
        L-----12 [twelve]
            L-----11 [eleven]
        R-----99 [ninety-nine]
R-----107 [one hundred seven]
    L-----101 [one hundred one]
    R-----1001 [one thousand one]
        R-----1010 [one thousand ten]

m.begin():
ptr=0x005BB4A0 Left=0x005BB3A0 Parent=0x005BB4C0 Right=0
    ↴ x005BB3A0 Color=1 Isnil=0
first=0 second=[zero]
m.end():
ptr=0x005BB3A0 Left=0x005BB4A0 Parent=0x005BB3C0 Right=0
    ↴ x005BB580 Color=1 Isnil=1

dumping s as set:
ptr=0x0020FDFC, Myhead=0x005BB5E0, Mysize=6
ptr=0x005BB5E0 Left=0x005BB640 Parent=0x005BB600 Right=0
    ↴ x005BB6A0 Color=1 Isnil=1
ptr=0x005BB600 Left=0x005BB660 Parent=0x005BB5E0 Right=0
    ↴ x005BB620 Color=1 Isnil=0
first=123
ptr=0x005BB660 Left=0x005BB640 Parent=0x005BB600 Right=0
    ↴ x005BB680 Color=1 Isnil=0
first=12
ptr=0x005BB640 Left=0x005BB5E0 Parent=0x005BB660 Right=0
    ↴ x005BB5E0 Color=0 Isnil=0
first=11
ptr=0x005BB680 Left=0x005BB5E0 Parent=0x005BB660 Right=0
    ↴ x005BB5E0 Color=0 Isnil=0
first=100
ptr=0x005BB620 Left=0x005BB5E0 Parent=0x005BB600 Right=0
    ↴ x005BB6A0 Color=1 Isnil=0
first=456
ptr=0x005BB6A0 Left=0x005BB5E0 Parent=0x005BB620 Right=0
    ↴ x005BB5E0 Color=0 Isnil=0
first=1001
As a tree:
root---123
    L-----12
        L-----11

```

```

R-----100
R-----456
R-----1001
s.begin():
ptr=0x005BB640 Left=0x005BB5E0 Parent=0x005BB660 Right=0x005BB6A0
    ↴ x005BB5E0 Color=0 Isnil=0
first=11
s.end():
ptr=0x005BB5E0 Left=0x005BB640 Parent=0x005BB600 Right=0x005BB6A0
    ↴ x005BB6A0 Color=1 Isnil=1

```

.

std::map, first y second std::pair std::set .

(51.4.2 on page 740).

std::list, . .begin()..operator-- y operator++ . [Cor+09].
.end() «landing zone» en HDD⁴.

GCC

```

#include <stdio.h>
#include <map>
#include <set>
#include <string>
#include <iostream>

struct map_pair
{
    int key;
    const char *value;
};

struct tree_node
{
    int M_color; // 0 - Red, 1 - Black
    struct tree_node *M_parent;
    struct tree_node *M_left;
    struct tree_node *M_right;
};

struct tree_struct
{
    int M_key_compare;

```

⁴Hard disk drive


```

{
    struct map_pair *p=(struct map_pair *) ↵
↳ point_after_struct;
    printf ("%d [%s]\n", p->key, p->value);
}

if (n->M_left)
{
    printf ("%.*sL-----", tabs, ALOT_OF_TABS);
    dump_as_tree (tabs+1, n->M_left, is_set);
};

if (n->M_right)
{
    printf ("%.*sR-----", tabs, ALOT_OF_TABS);
    dump_as_tree (tabs+1, n->M_right, is_set);
};

void dump_map_and_set(struct tree_struct *m, bool is_set)
{
    printf ("ptr=0x%p, M_key_compare=0x%xx, M_header=0x%p, ↵
↳ M_node_count=%d\n",
           m, m->M_key_compare, &m->M_header, m->M_node_count);
    dump_tree_node (m->M_header.M_parent, is_set, true, true);
    printf ("As a tree:\n");
    printf ("root----");
    dump_as_tree (1, m->M_header.M_parent, is_set);
};

int main()
{
    // map

    std::map<int, const char*> m;

    m[10]="ten";
    m[20]="twenty";
    m[3]="three";
    m[101]="one hundred one";
    m[100]="one hundred";
    m[12]="twelve";
    m[107]="one hundred seven";
    m[0]="zero";
    m[1]="one";
    m[6]="six";
    m[99]="ninety-nine";
    m[5]="five";
}

```

```

m[11]="eleven";
m[1001]="one thousand one";
m[1010]="one thousand ten";
m[2]="two";
m[9]="nine";

printf ("dumping m as map:\n");
dump_map_and_set ((struct tree_struct *)(void*)&m, false);

std::map<int, const char*>::iterator it1=m.begin();
printf ("m.begin():\n");
dump_tree_node ((struct tree_node *)*(void**)&it1, false, ↴
↳ false, true);
it1=m.end();
printf ("m.end():\n");
dump_tree_node ((struct tree_node *)*(void**)&it1, false, ↴
↳ false, false);

// set

std::set<int> s;
s.insert(123);
s.insert(456);
s.insert(11);
s.insert(12);
s.insert(100);
s.insert(1001);
printf ("dumping s as set:\n");
dump_map_and_set ((struct tree_struct *)(void*)&s, true);
std::set<int>::iterator it2=s.begin();
printf ("s.begin():\n");
dump_tree_node ((struct tree_node *)*(void**)&it2, true, ↴
↳ false, true);
it2=s.end();
printf ("s.end():\n");
dump_tree_node ((struct tree_node *)*(void**)&it2, true, ↴
↳ false, false);
};

dumping m as map:
ptr=0x0028FE3C, M_key_compare=0x402b70, M_header=0x0028FE40, ↴
↳ M_node_count=17
ptr=0x007A4988 M_left=0x007A4C00 M_parent=0x0028FE40 M_right=0x007A4B80, ↴
↳ M_color=1
key=10 value=[ten]

```

Listing 51.36: GCC 4.8.1

```

ptr=0x007A4C00 M_left=0x007A4BE0 M_parent=0x007A4988 M_right=0x
    ↴ x007A4C60 M_color=1
key=1 value=[one]
ptr=0x007A4BE0 M_left=0x00000000 M_parent=0x007A4C00 M_right=0x
    ↴ x00000000 M_color=1
key=0 value=[zero]
ptr=0x007A4C60 M_left=0x007A4B40 M_parent=0x007A4C00 M_right=0x
    ↴ x007A4C20 M_color=0
key=5 value=[five]
ptr=0x007A4B40 M_left=0x007A4CE0 M_parent=0x007A4C60 M_right=0x
    ↴ x00000000 M_color=1
key=3 value=[three]
ptr=0x007A4CE0 M_left=0x00000000 M_parent=0x007A4B40 M_right=0x
    ↴ x00000000 M_color=0
key=2 value=[two]
ptr=0x007A4C20 M_left=0x00000000 M_parent=0x007A4C60 M_right=0x
    ↴ x007A4D00 M_color=1
key=6 value=[six]
ptr=0x007A4D00 M_left=0x00000000 M_parent=0x007A4C20 M_right=0x
    ↴ x00000000 M_color=0
key=9 value=[nine]
ptr=0x007A4B80 M_left=0x007A49A8 M_parent=0x007A4988 M_right=0x
    ↴ x007A4BC0 M_color=1
key=100 value=[one hundred]
ptr=0x007A49A8 M_left=0x007A4BA0 M_parent=0x007A4B80 M_right=0x
    ↴ x007A4C40 M_color=0
key=20 value=[twenty]
ptr=0x007A4BA0 M_left=0x007A4C80 M_parent=0x007A49A8 M_right=0x
    ↴ x00000000 M_color=1
key=12 value=[twelve]
ptr=0x007A4C80 M_left=0x00000000 M_parent=0x007A4BA0 M_right=0x
    ↴ x00000000 M_color=0
key=11 value=[eleven]
ptr=0x007A4C40 M_left=0x00000000 M_parent=0x007A49A8 M_right=0x
    ↴ x00000000 M_color=1
key=99 value=[ninety-nine]
ptr=0x007A4BC0 M_left=0x007A4B60 M_parent=0x007A4B80 M_right=0x
    ↴ x007A4CA0 M_color=0
key=107 value=[one hundred seven]
ptr=0x007A4B60 M_left=0x00000000 M_parent=0x007A4BC0 M_right=0x
    ↴ x00000000 M_color=1
key=101 value=[one hundred one]
ptr=0x007A4CA0 M_left=0x00000000 M_parent=0x007A4BC0 M_right=0x
    ↴ x007A4CC0 M_color=1
key=1001 value=[one thousand one]
ptr=0x007A4CC0 M_left=0x00000000 M_parent=0x007A4CA0 M_right=0x
    ↴ x00000000 M_color=0

```

```

key=1010 value=[one thousand ten]
As a tree:
root----10 [ten]
    -----1 [one]
        -----0 [zero]
        -----5 [five]
            -----3 [three]
                -----2 [two]
            -----6 [six]
                -----9 [nine]
    -----100 [one hundred]
        -----20 [twenty]
            -----12 [twelve]
                -----11 [eleven]
            -----99 [ninety-nine]
    -----107 [one hundred seven]
        -----101 [one hundred one]
        -----1001 [one thousand one]
                    -----1010 [one thousand ten]

m.begin():
ptr=0x007A4BE0 M_left=0x00000000 M_parent=0x007A4C00 M_right=0x00000000
    ↴ x00000000 M_color=1
key=0 value=[zero]
m.end():
ptr=0x0028FE40 M_left=0x007A4BE0 M_parent=0x007A4988 M_right=0x00000000
    ↴ x007A4CC0 M_color=0

dumping s as set:
ptr=0x0028FE20, M_key_compare=0x8, M_header=0x0028FE24, ↴
    ↴ M_node_count=6
ptr=0x007A1E80 M_left=0x01D5D890 M_parent=0x0028FE24 M_right=0x00000000
    ↴ x01D5D850 M_color=1
key=123
ptr=0x01D5D890 M_left=0x01D5D870 M_parent=0x007A1E80 M_right=0x00000000
    ↴ x01D5D8B0 M_color=1
key=12
ptr=0x01D5D870 M_left=0x00000000 M_parent=0x01D5D890 M_right=0x00000000
    ↴ x00000000 M_color=0
key=11
ptr=0x01D5D8B0 M_left=0x00000000 M_parent=0x01D5D890 M_right=0x00000000
    ↴ x00000000 M_color=0
key=100
ptr=0x01D5D850 M_left=0x00000000 M_parent=0x007A1E80 M_right=0x00000000
    ↴ x01D5D8D0 M_color=1
key=456
ptr=0x01D5D8D0 M_left=0x00000000 M_parent=0x01D5D850 M_right=0x00000000
    ↴ x00000000 M_color=0

```

```

key=1001
As a tree:
root----123
    L-----12
        L-----11
        R-----100
    R-----456
        R-----1001
s.begin():
ptr=0x01D5D870 M_left=0x00000000 M_parent=0x01D5D890 M_right=0x00000000
    ↴ x00000000 M_color=0
key=11
s.end():
ptr=0x0028FE24 M_left=0x01D5D870 M_parent=0x007A1E80 M_right=0x00000000
    ↴ x01D5D8D0 M_color=0

```

⁵.,

(GCC)

Listing 51.37: GCC

```

#include <stdio.h>
#include <map>
#include <set>
#include <string>
#include <iostream>

struct map_pair
{
    int key;
    const char *value;
};

struct tree_node
{
    int M_color; // 0 - Red, 1 - Black
    struct tree_node *M_parent;
    struct tree_node *M_left;
    struct tree_node *M_right;
};

struct tree_struct
{
    int M_key_compare;

```

⁵<http://go.yurichev.com/17084>


```
    dump_map_and_set ((struct tree_struct *)(void*)&s);
    s.insert(667);
    s.insert(1);
    s.insert(4);
    s.insert(7);
    printf ("\n");
    printf ("667, 1, 4, 7 are inserted\n");
    dump_map_and_set ((struct tree_struct *)(void*)&s);
    printf ("\n");
}
};
```

Listing 51.38: GCC 4.8.1

```
123, 456 are inserted
root----123
        R-----456

11, 12 are inserted
root----123
        L-----11
                R-----12
        R-----456

100, 1001 are inserted
root----123
        L-----12
                L-----11
                R-----100
        R-----456
                R-----1001

667, 1, 4, 7 are inserted
root----12
        L-----4
                L-----1
                R-----11
                        L-----7
        R-----123
                L-----100
                R-----667
                        L-----456
                        R-----1001
```

Capítulo 52

, array[-1].

. :

```
#include <stdio.h>

int main()
{
    int random_value=0x11223344;
    unsigned char array[10];
    int i;
    unsigned char *fakearray=&array[-1];

    for (i=0; i<10; i++)
        array[i]=i;

    printf ("first element %d\n", fakearray[1]);
    printf ("second element %d\n", fakearray[2]);
    printf ("last element %d\n", fakearray[10]);

    printf ("array[-1]=%02X, array[-2]=%02X, array[-3]=%02X,\n",
    ↴ , array[-4]=%02X\n",
            array[-1],
            array[-2],
            array[-3],
            array[-4]);
}
```

Listing 52.1: Sin optimización MSVC 2010

1	\$SG2751 DB	'first element %d', 0aH, 00H
2	\$SG2752 DB	'second element %d', 0aH, 00H
3	\$SG2753 DB	'last element %d', 0aH, 00H

```

5 $SG2754 DB      'array[-1]=%02X, array[-2]=%02X, array[-3]=%02X'
   ↳ , array[-4']
   DB      ']=%02X', 0aH, 00H
7
8 _fakearray$ = -24           ; size = 4
9 _random_value$ = -20       ; size = 4
10 _array$ = -16            ; size = 10
11 _i$ = -4                ; size = 4
12 _main    PROC
13     push    ebp
14     mov     ebp, esp
15     sub     esp, 24
16     mov     DWORD PTR _random_value$[ebp], 287454020 ; ↳
   ↳ 11223344H
17     ;
18     lea     eax, DWORD PTR _array$[ebp]
19     add     eax, -1 ; eax=eax-1
20     mov     DWORD PTR _fakearray$[ebp], eax
21     mov     DWORD PTR _i$[ebp], 0
22     jmp     SHORT $LN3@main
23     ;
24 $LN2@main:
25     mov     ecx, DWORD PTR _i$[ebp]
26     add     ecx, 1
27     mov     DWORD PTR _i$[ebp], ecx
28 $LN3@main:
29     cmp     DWORD PTR _i$[ebp], 10
30     jge     SHORT $LN1@main
31     mov     edx, DWORD PTR _i$[ebp]
32     mov     al, BYTE PTR _i$[ebp]
33     mov     BYTE PTR _array$[ebp+edx], al
34     jmp     SHORT $LN2@main
35 $LN1@main:
36     mov     ecx, DWORD PTR _fakearray$[ebp]
37     ; ecx= fakearray[0], ecx+1 fakearray[1]  array[0]
38     movzx  edx, BYTE PTR [ecx+1]
39     push   edx
40     push   OFFSET $SG2751 ; 'first element %d'
41     call   _printf
42     add    esp, 8
43     mov    eax, DWORD PTR _fakearray$[ebp]
44     ; eax= fakearray[0], eax+2 fakearray[2]  array[1]
45     movzx  ecx, BYTE PTR [eax+2]
46     push   ecx
47     push   OFFSET $SG2752 ; 'second element %d'
48     call   _printf
49     add    esp, 8

```

```

50      mov     edx, DWORD PTR _fakearray$[ebp]
51      ; edx= fakearray[0], edx+10  fakearray[10]  array[9]
52      movzx   eax, BYTE PTR [edx+10]
53      push    eax
54      push    OFFSET $SG2753 ; 'last element %d'
55      call    _printf
56      add    esp, 8
57      ; 4, 3, 2 1  array[0]  array[]
58      lea     ecx, DWORD PTR _array$[ebp]
59      movzx   edx, BYTE PTR [ecx-4]
60      push    edx
61      lea     eax, DWORD PTR _array$[ebp]
62      movzx   ecx, BYTE PTR [eax-3]
63      push    ecx
64      lea     edx, DWORD PTR _array$[ebp]
65      movzx   eax, BYTE PTR [edx-2]
66      push    eax
67      lea     ecx, DWORD PTR _array$[ebp]
68      movzx   edx, BYTE PTR [ecx-1]
69      push    edx
70      push    OFFSET $SG2754 ; 'array[-1]=%02X, array[-2]=%02X'
71      ↳ X, array[-3]=%02X, array[-4]=%02X'
72      call    _printf
73      add    esp, 20
74      xor    eax, eax
75      mov    esp, ebp
76      pop    ebp
77      ret    0
_main ENDP

```

fakearray[1] array[0].array[]? random_value array[] 0x11223344.

.

:

first element 0
second element 1
last element 9
array[-1]=11, array[-2]=22, array[-3]=33, array[-4]=44

Listing 52.2: Sin optimización MSVC 2010

CPU Stack	
Address	Value
001DFBCC	/001DFBD3 ;
001DFBD0	11223344 ; random_value
001DFBD4	03020100 ; 4 array[]

001DFBD8	07060504 ; 4 array[]
001DFBDC	00CB0908 ; array[]
001DFBE0	0000000A ;
001DFBE4	001DFC2C ;
001DFBE8	\ 00CB129D ;

fakearray[] (0x001DFBD3) array[] (0x001DFBD4), .

Capítulo 53

Windows 16-bit

3.11. 96/98/ME [Windows NT](#). [Windows NT](#) .

OpenWatcom 1.9 :

```
wcl.exe -i=C:/WATCOM/h/win/ -s -os -bt=windows -bcl=windows
example.c
```

53.1 Ejemplo#1

```
#include <windows.h>

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    MessageBeep(MB_ICONEXCLAMATION);
    return 0;
};
```

```
WinMain      proc near
             push   bp
             mov    bp, sp
             mov    ax, 30h ; '0'    ; MB_ICONEXCLAMATION
             ↳ constant
             push   ax
             call   MESSAGEBEEP
             xor    ax, ax           ; return 0
```

WinMain	pop bp retn 0Ah endp
---------	----------------------------

53.2 Ejemplo #2

```
#include <windows.h>

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    MessageBox (NULL, "hello, world", "caption", ↴
    ↴ MB_YESNOCANCEL);
    return 0;
}
```

```
WinMain      proc near
push    bp
mov     bp, sp
xor    ax, ax          ; NULL
push    ax
push    ds
mov     ax, offset aHelloWorld ; 0x18. "hello, ↴
    ↴ world"
push    ax
push    ds
mov     ax, offset aCaption ; 0x10. "caption"
push    ax
mov     ax, 3           ; MB_YESNOCANCEL
push    ax
call    MESSAGEBOX
xor    ax, ax          ; return 0
pop     bp
retn    0Ah
WinMain      endp

dseg02:0010 aCaption        db 'caption',0
dseg02:0018 aHelloWorld     db 'hello, world',0
```

(MB_YESNOCANCEL), (NULL). **stack pointer:** RETN 0Ah . stdcall (64.2 on page 869), .

53.3 Ejemplo #3

```
#include <windows.h>

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    int result=MessageBox (NULL, "hello, world", "caption",
                         MB_YESNOCANCEL);

    if (result==IDCANCEL)
        MessageBox (NULL, "you pressed cancel", "caption",
                    MB_OK);
    else if (result==IDYES)
        MessageBox (NULL, "you pressed yes", "caption", MB_OK);
    else if (result==IDNO)
        MessageBox (NULL, "you pressed no", "caption", MB_OK);

    return 0;
}
```

<pre>WinMain proc near push bp mov bp, sp xor ax, ax ; NULL push ax push ds mov ax, offset aHelloWorld ; "hello, world" push ax push ds mov ax, offset aCaption ; "caption" push ax mov ax, 3 ; MB_YESNOCANCEL push ax call MESSAGEBOX cmp ax, 2 ; IDCANCEL jnz short loc_2F xor ax, ax push ax push ds mov ax, offset aYouPressedCanc ; "you ↴ ↴ pressed cancel"</pre>

```

        jmp    short loc_49
loc_2F:
        cmp    ax, 6           ; IDYES
        jnz    short loc_3D
        xor    ax, ax
        push   ax
        push   ds
        mov    ax, offset aYouPressedYes ; "you ↴
    ↴ pressed yes"
        jmp    short loc_49
loc_3D:
        cmp    ax, 7           ; IDNO
        jnz    short loc_57
        xor    ax, ax
        push   ax
        push   ds
        mov    ax, offset aYouPressedNo ; "you pressed ↴
    ↴ no"
loc_49:
        push   ax
        push   ds
        mov    ax, offset aCaption ; "caption"
        push   ax
        xor    ax, ax
        push   ax
        call   MESSAGEBOX
loc_57:
        xor    ax, ax
        pop    bp
        retn  0Ah
WinMain
        endp

```

53.4 Ejemplo #4

```

#include <windows.h>

int PASCAL func1 (int a, int b, int c)
{
    return a*b+c;
};

long PASCAL func2 (long a, long b, long c)
{
    return a*b+c;
}

```

```

};;

long PASCAL func3 ( long a, long b, long c, int d)
{
    return a*b+c-d;
};

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    func1 (123, 456, 789);
    func2 (600000, 700000, 800000);
    func3 (600000, 700000, 800000, 123);
    return 0;
};

```

```

func1      proc near

c          = word ptr 4
b          = word ptr 6
a          = word ptr 8

    push    bp
    mov     bp, sp
    mov     ax, [bp+a]
    imul   [bp+b]
    add    ax, [bp+c]
    pop    bp
    retn   6
func1      endp

func2      proc near

arg_0      = word ptr 4
arg_2      = word ptr 6
arg_4      = word ptr 8
arg_6      = word ptr 0Ah
arg_8      = word ptr 0Ch
arg_A      = word ptr 0Eh

    push    bp
    mov     bp, sp
    mov     ax, [bp+arg_8]
    mov     dx, [bp+arg_A]
    mov     bx, [bp+arg_4]

```

```

        mov    cx, [bp+arg_6]
        call   sub_B2 ; long 32-bit multiplication
        add    ax, [bp+arg_0]
        adc    dx, [bp+arg_2]
        pop    bp
        retn  12
func2      endp

func3      proc near

arg_0       = word ptr 4
arg_2       = word ptr 6
arg_4       = word ptr 8
arg_6       = word ptr 0Ah
arg_8       = word ptr 0Ch
arg_A       = word ptr 0Eh
arg_C       = word ptr 10h

        push   bp
        mov    bp, sp
        mov    ax, [bp+arg_A]
        mov    dx, [bp+arg_C]
        mov    bx, [bp+arg_6]
        mov    cx, [bp+arg_8]
        call   sub_B2 ; long 32-bit multiplication
        mov    cx, [bp+arg_2]
        add    cx, ax
        mov    bx, [bp+arg_4]
        adc    bx, dx      ; BX=high part, CX=low ↴
        ↵ part
        mov    ax, [bp+arg_0]
        cwd
        ↵ high part d
        mov    cx, ax
        sub    cx, cx
        sbb    bx, dx
        mov    dx, bx
        pop    bp
        retn  14
func3      endp

WinMain    proc near
        push   bp
        mov    bp, sp
        mov    ax, 123
        push   ax
        mov    ax, 456

```

```

push    ax
mov     ax, 789
push    ax
call   func1
mov     ax, 9      ; high part of 600000
push    ax
mov     ax, 27C0h ; low part of 600000
push    ax
mov     ax, 0Ah    ; high part of 700000
push    ax
mov     ax, 0AE60h ; low part of 700000
push    ax
mov     ax, 0Ch    ; high part of 800000
push    ax
mov     ax, 3500h ; low part of 800000
push    ax
call   func2
mov     ax, 9      ; high part of 600000
push    ax
mov     ax, 27C0h ; low part of 600000
push    ax
mov     ax, 0Ah    ; high part of 700000
push    ax
mov     ax, 0AE60h ; low part of 700000
push    ax
mov     ax, 0Ch    ; high part of 800000
push    ax
mov     ax, 3500h ; low part of 800000
push    ax
mov     ax, 7Bh    ; 123
push    ax
call   func3
xor    ax, ax      ; return 0
pop    bp
retn  0Ah
WinMain endp

```

. (24 on page 519).

sub_B2 here «long multiplication», . :E on page 1254, D on page 1253.

ADD/ADC :

SUB/SBB :

.

WinMain().

53.5 Ejemplo #5

```
#include <windows.h>

int PASCAL string_compare (char *s1, char *s2)
{
    while (1)
    {
        if (*s1!=*s2)
            return 0;
        if (*s1==0 || *s2==0)
            return 1; // end of string
        s1++;
        s2++;
    };
};

int PASCAL string_compare_far (char far *s1, char far *s2)
{
    while (1)
    {
        if (*s1!=*s2)
            return 0;
        if (*s1==0 || *s2==0)
            return 1; // end of string
        s1++;
        s2++;
    };
};

void PASCAL remove_digits (char *s)
{
    while (*s)
    {
        if (*s>='0' && *s<='9')
            *s='-';
        s++;
    };
};

char str[]="hello 1234 world";
```

```

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    string_compare ("asd", "def");
    string_compare_far ("asd", "def");
    remove_digits (str);
    MessageBox (NULL, str, "caption", MB_YESNOCANCEL);
    return 0;
}

```

```

string_compare proc near

arg_0 = word ptr 4
arg_2 = word ptr 6

    push    bp
    mov     bp, sp
    push    si
    mov     si, [bp+arg_0]
    mov     bx, [bp+arg_2]

loc_12: ; CODE XREF: string_compare+21j
    mov     al, [bx]
    cmp     al, [si]
    jz      short loc_1C
    xor     ax, ax
    jmp     short loc_2B

loc_1C: ; CODE XREF: string_compare+Ej
    test    al, al
    jz      short loc_22
    jnz    short loc_27

loc_22: ; CODE XREF: string_compare+16j
    mov     ax, 1
    jmp     short loc_2B

loc_27: ; CODE XREF: string_compare+18j
    inc     bx
    inc     si
    jmp     short loc_12

```

```

loc_2B: ; CODE XREF: string_compare+12j
        ; string_compare+1Dj
    pop     si
    pop     bp
    retn     4
string_compare    endp

string_compare_far proc near ; CODE XREF: WinMain+18p

arg_0 = word ptr 4
arg_2 = word ptr 6
arg_4 = word ptr 8
arg_6 = word ptr 0Ah

    push     bp
    mov      bp, sp
    push     si
    mov      si, [bp+arg_0]
    mov      bx, [bp+arg_4]

loc_3A: ; CODE XREF: string_compare_far+35j
    mov      es, [bp+arg_6]
    mov      al, es:[bx]
    mov      es, [bp+arg_2]
    cmp      al, es:[si]
    jz       short loc_4C
    xor      ax, ax
    jmp     short loc_67

loc_4C: ; CODE XREF: string_compare_far+16j
    mov      es, [bp+arg_6]
    cmp      byte ptr es:[bx], 0
    jz       short loc_5E
    mov      es, [bp+arg_2]
    cmp      byte ptr es:[si], 0
    jnz     short loc_63

loc_5E: ; CODE XREF: string_compare_far+23j
    mov      ax, 1
    jmp     short loc_67

loc_63: ; CODE XREF: string_compare_far+2Cj
    inc      bx
    inc      si
    jmp     short loc_3A

```

```
loc_67: ; CODE XREF: string_compare_far+1Aj
        ; string_compare_far+31j
    pop     si
    pop     bp
    retn     8
string_compare_far endp

remove_digits proc near ; CODE XREF: WinMain+1Fp
arg_0 = word ptr 4

    push     bp
    mov      bp, sp
    mov      bx, [bp+arg_0]

loc_72: ; CODE XREF: remove_digits+18j
    mov      al, [bx]
    test    al, al
    jz      short loc_86
    cmp      al, 30h ; '0'
    jb       short loc_83
    cmp      al, 39h ; '9'
    ja       short loc_83
    mov      byte ptr [bx], 2Dh ; '-'

loc_83: ; CODE XREF: remove_digits+Ej
        ; remove_digits+12j
    inc      bx
    jmp      short loc_72

loc_86: ; CODE XREF: remove_digits+Aj
    pop     bp
    retn     2
remove_digits endp

WinMain proc near ; CODE XREF: start+EDp
    push     bp
    mov      bp, sp
    mov      ax, offset aAsd ; "asd"
    push     ax
    mov      ax, offset aDef ; "def"
    push     ax
    call    string_compare
    push     ds
```

```

    mov      ax, offset aAsd ; "asd"
    push     ax
    push     ds
    mov      ax, offset aDef ; "def"
    push     ax
    call    string_compare_far
    mov      ax, offset aHello1234World ; "hello 1234 world"
    push     ax
    call    remove_digits
    xor      ax, ax
    push     ax
    push     ds
    mov      ax, offset aHello1234World ; "hello 1234 world"
    push     ax
    push     ds
    mov      ax, offset aCaption ; "caption"
    push     ax
    mov      ax, 3 ; MB_YESNOCANCEL
    push     ax
    call    MESSAGEBOX
    xor      ax, ax
    pop      bp
    retn    0Ah
WinMain endp

```

«near» «far» .

[: 94 on page 1160.](#)

«near» . string_compare() (mov al, [bx] mov al, ds:[bx]DS).

«far» . string_compare_far() (mov al, es:[bx]). «far» MessageBox(): [53.2 on page 777..](#)

.

.

«memory model».

53.6 Ejemplo #6

```
#include <windows.h>
#include <time.h>
#include <stdio.h>
```

```

char strbuf[256];

int PASCAL WinMain( HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow )
{
    struct tm *t;
    time_t unix_time;

    unix_time=time(NULL);

    t=localtime (&unix_time);

    sprintf (strbuf, "%04d-%02d-%02d %02d:%02d:%02d", t->tm_year+1900, t->tm_mon, t->tm_mday,
             t->tm_hour, t->tm_min, t->tm_sec);

    MessageBox (NULL, strbuf, "caption", MB_OK);
    return 0;
}

```

WinMain	proc near
var_4	= word ptr -4
var_2	= word ptr -2
	push bp
	mov bp, sp
	push ax
	push ax
	xor ax, ax
	call time_
↳ time	mov [bp+var_4], ax ; low part of UNIX ↴
↳ time	mov [bp+var_2], dx ; high part of UNIX ↴
↳ high part	lea ax, [bp+var_4] ; take a pointer of ↴
	call localtime_
	mov bx, ax ; t
	push word ptr [bx] ; second
	push word ptr [bx+2] ; minute
	push word ptr [bx+4] ; hour
	push word ptr [bx+6] ; day
	push word ptr [bx+8] ; month

```

        mov     ax, [bx+0Ah]      ; year
        add     ax, 1900
        push    ax
        mov     ax, offset a04d02d02d02d02 ; "%04d-%02d%02d %02d"
        ↳ -%02d %02d:%02d:%02d"
        push    ax
        mov     ax, offset strbuf
        push    ax
        call    sprintf_
        add     sp, 10h
        xor     ax, ax           ; NULL
        push    ax
        push    ds
        mov     ax, offset strbuf
        push    ax
        push    ds
        mov     ax, offset aCaption ; "caption"
        push    ax
        xor     ax, ax           ; MB_OK
        push    ax
        call    MESSAGEBOX
        xor     ax, ax
        mov     sp, bp
        pop    bp
        retn   0Ah
WinMain    endp

```

. localtime(). localtime() struct tm ..
time() y localtime() AX, DX, BX y CX, ..
sprintf() .

53.6.1

```

:
#include <windows.h>
#include <time.h>
#include <stdio.h>

char strbuf[256];
struct tm *t;
time_t unix_time;

int PASCAL WinMain( HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine,

```

```

        int nCmdShow )
{

    unix_time=time(NULL);

    t=localtime (&unix_time);

    sprintf (strbuf, "%04d-%02d-%02d %02d:%02d:%02d", t->tm_year+1900, t->tm_mon, t->tm_mday,
             t->tm_hour, t->tm_min, t->tm_sec);

    MessageBox (NULL, strbuf, "caption", MB_OK);
    return 0;
};

```

```

unix_time_low    dw 0
unix_time_high   dw 0
t                 dw 0

WinMain          proc near
    push    bp
    mov     bp, sp
    xor    ax, ax
    call   time_
    mov    unix_time_low, ax
    mov    unix_time_high, dx
    mov    ax, offset unix_time_low
    call   localtime_
    mov    bx, ax
    mov    t, ax           ; will not be used
    ; in future...
    push    word ptr [bx]      ; seconds
    push    word ptr [bx+2]    ; minutes
    push    word ptr [bx+4]    ; hour
    push    word ptr [bx+6]    ; day
    push    word ptr [bx+8]    ; month
    mov    ax, [bx+0Ah]        ; year
    add    ax, 1900
    push    ax
    mov    ax, offset a04d02d02d02d02 ; "%04d-%02d-%02d %02d:%02d:%02d"
    push    ax
    mov    ax, offset strbuf
    push    ax
    call   sprintf_
    add    sp, 10h
    xor    ax, ax           ; NULL

```

```
push    ax
push    ds
mov     ax, offset strbuf
push    ax
push    ds
mov     ax, offset aCaption ; "caption"
push    ax
xor    ax, ax           ; MB_OK
push    ax
call   MESSAGEBOX
xor    ax, ax           ; return 0
pop    bp
retn  0Ah
WinMain endp
```

Parte IV

Java

Capítulo 54

Java

54.1

•
•
•
•
• .
• .
• .
• .

:javap -c -verbose
:[Jav13].

54.2

```
public class ret
{
    public static int main(String[] args)
    {
        return 0;
}
```

:

```
javac ret.java
```

...:

```
javap -c -verbose ret.class
```

:

Listing 54.1: JDK 1.7 (excerpt)

```
public static int main(java.lang.String[]);
  flags: ACC_PUBLIC, ACC_STATIC
  Code:
    stack=1, locals=1, args_size=1
      0:  iconst_0
      1:  ireturn
```

1. iconst_m1 -1.

Stack is used in JVM for data passing into functions to be called and also returning values. So `iconst_0` pushed 0 into stack. `ireturn` returns integer value (*i* in name mean *integer*) from the [TOS²](#).

1234:

```
public class ret
{
    public static int main(String[] args)
    {
        return 1234;
    }
}
```

...:

Listing 54.2: JDK 1.7 (excerpt)

```
public static int main(java.lang.String[]);
  flags: ACC_PUBLIC, ACC_STATIC
  Code:
    stack=1, locals=1, args_size=1
      0:  sipush      1234
      3:  ireturn
```

¹: [3.5.2 on page 26](#).²Top Of Stack

```
sipush (short integer) 1234 . short .
```

```
public class ret
{
    public static int main(String[] args)
    {
        return 12345678;
    }
}
```

Listing 54.3: Constant pool

```
...
#2 = Integer           12345678
...
```

```
public static int main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
0: ldc             #2           // int 12345678
2: ireturn
```

[28.3 on page 575](#), [29.1 on page 580](#).

!

Boolean:

```
public class ret
{
    public static boolean main(String[] args)
    {
        return true;
    }
}
```

```
public static boolean main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
0: iconst_1
1: ireturn
```

short:

```
public class ret
{
```

```
public static short main(String[] args)
{
    return 1234;
}
```

```
public static short main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
0: sipush      1234
3: ireturn
```

...y char!

```
public class ret
{
    public static char main(String[] args)
    {
        return 'A';
    }
}
```

```
public static char main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
0: bipush      65
2: ireturn
```

bipush «push byte».

byte:

```
public class retc
{
    public static byte main(String[] args)
    {
        return 123;
    }
}
```

```
public static byte main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
```

```
0: bipush      123
2: ireturn
```

```
public class ret3
{
    public static long main(String[] args)
    {
        return 1234567890123456789L;
    }
}
```

Listing 54.4: Constant pool

```
...
#2 = Long          12345678901234567891
...
```

```
public static long main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
0: ldc2_w          #2                      // long ↴
↳ 12345678901234567891
3: lreturn
```

```
public class ret
{
    public static double main(String[] args)
    {
        return 123.456d;
    }
}
```

Listing 54.5: Constant pool

```
...
#2 = Double        123.456d
...
```

```
public static double main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
0: ldc2_w          #2                      // double 123.456 ↴
↳ d
3: dreturn
```

```
public class ret
{
    public static float main(String[] args)
    {
        return 123.456f;
    }
}
```

Listing 54.6: Constant pool

```
...
#2 = Float          123.456f
...
```

```
public static float main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
0: ldc           #2                      // float 123.456f
2: freturn
```

```
public class ret
{
    public static void main(String[] args)
    {
        return;
    }
}
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=0, locals=1, args_size=1
0: return
```

54.3

```
public class calc
{
    public static int half(int a)
    {
        return a/2;
```

```
}
```

```
public static int half(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
  0: iload_0
  1: iconst_2
  2: idiv
  3: ireturn
```

iload_0 iconst_2 .

```
+---+
TOS ->| 2 |
+---+
| a |
+---+
```

```
+-----+
TOS ->| result |
+-----+
```

ireturn .

```
public class calc
{
    public static double half_double(double a)
    {
        return a/2.0;
    }
}
```

Listing 54.7: Constant pool

```
...
#2 = Double          2.0d
...
```

```
public static double half_double(double);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=4, locals=2, args_size=1
  0: dload_0
  1: ldc2_w           #2                      // double 2.0d
```

```
4: ddiv
5: dreturn
```

```
public class calc
{
    public static int sum(int a, int b)
    {
        return a+b;
    }
}
```

```
public static int sum(int, int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=2, args_size=2
0: iload_0
1: iload_1
2: iadd
3: ireturn
```

```
+---+
TOS ->| b |
+---+
| a |
+---+
```

```
+-----+
TOS ->| result |
+-----+
```

```
public static long lsum(long a, long b)
{
    return a+b;
}
```

...:

```
public static long lsum(long, long);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=4, locals=4, args_size=2
0: lload_0
1: lload_2
2: ladd
3: lreturn
```

```
public class calc
{
    public static int mult_add(int a, int b, int c)
    {
        return a*b+c;
    }
}
```

```
public static int mult_add(int, int, int);
```

```
flags: ACC_PUBLIC, ACC_STATIC
```

```
Code:
```

```
stack=2, locals=3, args_size=3
 0: iload_0
 1: iload_1
 2: imul
 3: iload_2
 4: iadd
 5: ireturn
```

```
+-----+
TOS ->| product |
+-----+
```

iload_2:

```
+-----+
TOS ->|   c   |
+-----+
| product |
+-----+
```

54.4

JVM³.

:

- (LVA⁴). istore.
- .
-

³Java virtual machine

⁴(Java) Local Variable Array

54.5

```
public class HalfRandom
{
    public static double f()
    {
        return Math.random()/2;
    }
}
```

Listing 54.8: Constant pool

```
...
#2 = Methodref           #18.#19      //  java/lang/Math.
  ↴ random:()D
#3 = Double              2.0d
...
#12 = Utf8               ()D
...
#18 = Class              #22          //  java/lang/Math
#19 = NameAndType        #23:#12     //  random:()D
#22 = Utf8               java/lang/Math
#23 = Utf8               random
```

```
public static double f();
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=4, locals=0, args_size=0
  0: invokestatic #2                  // Method java/
  ↴ lang/Math.random:()D
  3: ldc2_w       #3                  // double 2.0d
  6: ddiv
  7: dreturn
```

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World");
    }
}
```

Listing 54.9: Constant pool

```
...
#2 = Fieldref           #16.#17      //  java/lang/System.
  ↴ out:Ljava/io/PrintStream;
```

```

#3 = String          #18           // Hello, World
#4 = Methodref      #19:#20       // java/io/⤦
↳ PrintStream.println:(Ljava/lang/String;)V
...
#16 = Class          #23           // java/lang/System
#17 = NameAndType   #24:#25       // out:Ljava/io/⤦
↳ PrintStream;
#18 = Utf8           Hello, World
#19 = Class          #26           // java/io/⤦
↳ PrintStream
#20 = NameAndType   #27:#28       // println:(Ljava/⤦
↳ lang/String;)V
...
#23 = Utf8           java/lang/System
#24 = Utf8           out
#25 = Utf8           Ljava/io/PrintStream;
#26 = Utf8           java/io/PrintStream
#27 = Utf8           println
#28 = Utf8           (Ljava/lang/String;)V
...

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=2, locals=1, args_size=1
    0: getstatic      #2           // Field java/⤦
    ↳ lang/System.out:Ljava/io/PrintStream;
      3: ldc           #3           // String Hello, ⤤
    ↳ World
      5: invokevirtual #4           // Method java/io/⤦
    ↳ /PrintStream.println:(Ljava/lang/String;)V
      8: return

```

5.

54.6 beep()

```

public static void main(String[] args)
{
    java.awt.Toolkit.getDefaultToolkit().beep();
}

```

```
public static void main(java.lang.String[]);
```

⁵: [51.3 on page 723](#).

```

flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=1, locals=1, args_size=1
    0: invokestatic #2                      // Method java/awt/Toolkit.<clinit>()
    ↳ awt/Toolkit.getDefaultToolkit:()Ljava/awt/Toolkit;
      3: invokevirtual #3                   // Method java/awt/Toolkit.beep:()V
    ↳ awt/Toolkit.beep:()V
    6: return

```

54.7

```

public class LCG
{
    public static int rand_state;

    public void my_srand (int init)
    {
        rand_state=init;
    }

    public static int RNG_a=1664525;
    public static int RNG_c=1013904223;

    public int my_rand ()
    {
        rand_state=rand_state*RNG_a;
        rand_state=rand_state+RNG_c;
        return rand_state & 0x7fff;
    }
}

```

?

```

static {};
flags: ACC_STATIC
Code:
  stack=1, locals=0, args_size=0
    0: ldc           #5                  // int 1664525
    2: putstatic     #3                  // Field RNG_a:I
    5: ldc           #6                  // int 1013904223
    7: putstatic     #4                  // Field RNG_c:I
   10: return

```

```

public void my_srand(int);
flags: ACC_PUBLIC

```

Code:

```

stack=1, locals=2, args_size=2
  0: iload_1
  1: putstatic      #2                      // Field ↗
↳ rand_state:I
  4: return

```

my_rand():

```

public int my_rand();
flags: ACC_PUBLIC
Code:
stack=2, locals=1, args_size=1
  0: getstatic      #2                      // Field ↗
↳ rand_state:I
  3: getstatic      #3                      // Field RNG_a:I
  6: imul
  7: putstatic      #2                      // Field ↗
↳ rand_state:I
  10: getstatic     #2                      // Field ↗
↳ rand_state:I
  13: getstatic     #4                      // Field RNG_c:I
  16: iadd
  17: putstatic      #2                      // Field ↗
↳ rand_state:I
  20: getstatic     #2                      // Field ↗
↳ rand_state:I
  23: sipush        32767
  26: iand
  27: ireturn

```

54.8

```

.
public class abs
{
    public static int abs(int a)
    {
        if (a<0)
            return -a;
        return a;
    }
}

```

```
public static int abs(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=1, locals=1, args_size=1
    0: iload_0
    1: ifge           7
    4: iload_0
    5: ineg
    6: ireturn
    7: iload_0
    8: ireturn
```

:

```
public static int min (int a, int b)
{
    if (a>b)
        return b;
    return a;
}
```

:

```
public static int min(int, int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=2, locals=2, args_size=2
    0: iload_0
    1: iload_1
    2: if_icmple     7
    5: iload_1
    6: ireturn
    7: iload_0
    8: ireturn
```

if_icmple .

```
public static int max (int a, int b)
{
    if (a>b)
        return a;
    return b;
}
```

```
public static int max(int, int);
flags: ACC_PUBLIC, ACC_STATIC
```

Code:

```

stack=2, locals=2, args_size=2
 0: iload_0
 1: iload_1
 2: if_icmple    7
 5: iload_0
 6: ireturn
 7: iload_1
 8: ireturn

```

:

```

public class cond
{
    public static void f(int i)
    {
        if (i<100)
            System.out.print("<100");
        if (i==100)
            System.out.print("==100");
        if (i>100)
            System.out.print(">100");
        if (i==0)
            System.out.print("==0");
    }
}

```

```

public static void f(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
 0: iload_0
 1: bipush      100
 3: if_icmpge   14
 6: getstatic   #2           // Field java/lang/System.out
 ↴ lang/System.out:Ljava/io/PrintStream;
 9: ldc         #3           // String <100
11: invokevirtual #4           // Method java/io/PrintStream.print:(Ljava/lang/String;)V
 ↴ /PrintStream.print:(Ljava/lang/String;)V
14: iload_0
15: bipush      100
17: if_icmpne   28
20: getstatic   #2           // Field java/lang/System.out
 ↴ lang/System.out:Ljava/io/PrintStream;
23: ldc         #5           // String ==100
25: invokevirtual #4           // Method java/io/PrintStream.print:(Ljava/lang/String;)V
 ↴ /PrintStream.print:(Ljava/lang/String;)V

```

```

28: iload_0
29: bipush      100
31: if_icmple   42
34: getstatic   #2           // Field java/
    ↳ lang/System.out:Ljava/io/PrintStream;
    37: ldc          #6           // String >100
    39: invokevirtual #4           // Method java/io/
    ↳ /PrintStream.print:(Ljava/lang/String;)V
    42: iload_0
    43: ifne        54
    46: getstatic   #2           // Field java/
    ↳ lang/System.out:Ljava/io/PrintStream;
    49: ldc          #7           // String ==0
    51: invokevirtual #4           // Method java/io/
    ↳ /PrintStream.print:(Ljava/lang/String;)V
    54: return

```

if_icmpge .

54.9

```

public class minmax
{
    public static int min (int a, int b)
    {
        if (a>b)
            return b;
        return a;
    }

    public static int max (int a, int b)
    {
        if (a>b)
            return a;
        return b;
    }

    public static void main(String[] args)
    {
        int a=123, b=456;
        int max_value=max(a, b);
        int min_value=min(a, b);
        System.out.println(min_value);
        System.out.println(max_value);
    }
}

```

```

public static void main(java.lang.String[]);
  flags: ACC_PUBLIC, ACC_STATIC
  Code:
    stack=2, locals=5, args_size=1
      0: bipush        123
      2: istore_1
      3: sipush        456
      6: istore_2
      7: iload_1
      8: iload_2
      9: invokestatic #2                      // Method max:(II)I
  ↳ )I
    12: istore_3
    13: iload_1
    14: iload_2
    15: invokestatic #3                      // Method min:(II)I
  ↳ )I
    18: istore        4
    20: getstatic     #4                      // Field java/.<clinit>
  ↳ lang/System.out:Ljava/io/PrintStream;
    23: iload        4
    25: invokevirtual #5                      // Method java/io/PrintStream.<init>()
  ↳ /PrintStream.println:(I)V
    28: getstatic     #4                      // Field java/.<clinit>
  ↳ lang/System.out:Ljava/io/PrintStream;
    31: iload_3
    32: invokevirtual #5                      // Method java/io/PrintStream.println(I)V
  ↳ /PrintStream.println:(I)V
    35: return

```

54.10

```

public static int set (int a, int b)
{
    return a | 1<<b;
}

public static int clear (int a, int b)
{
    return a & (~(1<<b));
}

```

```

public static int set(int, int);
  flags: ACC_PUBLIC, ACC_STATIC

```

```

Code:
stack=3, locals=2, args_size=2
 0: iload_0
 1: iconst_1
 2: iload_1
 3: ishl
 4: ior
 5: ireturn

public static int clear(int, int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=3, locals=2, args_size=2
 0: iload_0
 1: iconst_1
 2: iload_1
 3: ishl
 4: iconst_m1
 5: ixor
 6: iand
 7: ireturn

```

(A.6.2 on page 1236).

```

public static long lset (long a, int b)
{
    return a | 1<<b;
}

public static long lclear (long a, int b)
{
    return a & (~(1<<b));
}

```

```

public static long lset(long, int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=4, locals=3, args_size=2
 0: lload_0
 1: iconst_1
 2: iload_2
 3: ishl
 4: i2l
 5: lor
 6: lreturn

public static long lclear(long, int);

```

```

flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=4, locals=3, args_size=2
  0: lload_0
  1: iconst_1
  2: iload_2
  3: ishl
  4: iconst_m1
  5: ixor
  6: i2l
  7: land
  8: lreturn

```

54.11

```

public class Loop
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 10; i++)
        {
            System.out.println(i);
        }
    }
}

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=2, locals=2, args_size=1
  0: iconst_1
  1: istore_1
  2: iload_1
  3: bipush      10
  5: if_icmpgt   21
  8: getstatic    #2           // Field java/
  ↳ lang/System.out:Ljava/io/PrintStream;
  11: iload_1
  12: invokevirtual #3           // Method java/io/
  ↳ /PrintStream.println:(I)V
  15: iinc       1, 1
  18: goto       2
  21: return

```

```
public class Fibonacci
{
    public static void main(String[] args)
    {
        int limit = 20, f = 0, g = 1;

        for (int i = 1; i <= limit; i++)
        {
            f = f + g;
            g = f - g;
            System.out.println(f);
        }
    }
}
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=5, args_size=1
 0: bipush          20
 2: istore_1
 3: iconst_0
 4: istore_2
 5: iconst_1
 6: istore_3
 7: iconst_1
 8: istore          4
10: iload           4
12: iload_1
13: if_icmpgt      37
16: iload_2
17: iload_3
18: iadd
19: istore_2
20: iload_2
21: iload_3
22: isub
23: istore_3
24: getstatic      #2          // Field java/
↳ lang/System.out:Ljava/io/PrintStream;
 27: iload_2
 28: invokevirtual #3          // Method java/io/
↳ /PrintStream.println:(I)V
 31: iinc           4, 1
 34: goto           10
37: return
```

- 0 –
- 1 – *limit*, 20
- 2 – *f*
- 3 – *g*
- 4 – *i*

54.12 switch()

```
public static void f(int a)
{
    switch (a)
    {
        case 0: System.out.println("zero"); break;
        case 1: System.out.println("one\n"); break;
        case 2: System.out.println("two\n"); break;
        case 3: System.out.println("three\n"); break;
        case 4: System.out.println("four\n"); break;
        default: System.out.println("something unknown\n");
    }
}
```

```
:
public static void f(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
  0: iload_0
  1: tableswitch   { // 0 to 4
      0: 36
      1: 47
      2: 58
      3: 69
      4: 80
      default: 91
  }
  36: getstatic      #2                      // Field java/
    ↳ lang/System.out:Ljava/io/PrintStream;
    39: ldc            #3                      // String zero
    41: invokevirtual #4                      // Method java/io/
    ↳ /PrintStream.println:(Ljava/lang/String;)V
    44: goto          99
```

```

    47: getstatic      #2          // Field java/\v
    ↳ lang/System.out:Ljava/io/PrintStream;
        50: ldc           #5          // String one\n
        52: invokevirtual #4          // Method java/io/\v
    ↳ /PrintStream.println:(Ljava/lang/String;)V
        55: goto          99
        58: getstatic      #2          // Field java/\v
    ↳ lang/System.out:Ljava/io/PrintStream;
        61: ldc           #6          // String two\n
        63: invokevirtual #4          // Method java/io/\v
    ↳ /PrintStream.println:(Ljava/lang/String;)V
        66: goto          99
        69: getstatic      #2          // Field java/\v
    ↳ lang/System.out:Ljava/io/PrintStream;
        72: ldc           #7          // String three\n
        74: invokevirtual #4          // Method java/io/\v
    ↳ /PrintStream.println:(Ljava/lang/String;)V
        77: goto          99
        80: getstatic      #2          // Field java/\v
    ↳ lang/System.out:Ljava/io/PrintStream;
        83: ldc           #8          // String four\n
        85: invokevirtual #4          // Method java/io/\v
    ↳ /PrintStream.println:(Ljava/lang/String;)V
        88: goto          99
        91: getstatic      #2          // Field java/\v
    ↳ lang/System.out:Ljava/io/PrintStream;
        94: ldc           #9          // String \v
    ↳ something unknown\n
        96: invokevirtual #4          // Method java/io/\v
    ↳ /PrintStream.println:(Ljava/lang/String;)V
        99: return

```

54.13

54.13.1

```

public static void main(String[] args)
{
    int a[]={new int[10];
    for (int i=0; i<10; i++)
        a[i]=i;
    dump (a);
}

```

```
public static void main(java.lang.String[]);
```

```

flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=3, locals=3, args_size=1
    0: bipush      10
    2: newarray     int
    4: astore_1
    5: iconst_0
    6: istore_2
    7: iload_2
    8: bipush      10
    10: if_icmpge   23
    13: aload_1
    14: iload_2
    15: iload_2
    16: iastore
    17: iinc         2, 1
    20: goto        7
    23: aload_1
    24: invokestatic #4           // Method dump:([I)V
    ↳ I)V
    27: return

```

```

public static void dump(int a[])
{
    for (int i=0; i<a.length; i++)
        System.out.println(a[i]);
}

```

```

public static void dump(int[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=3, locals=2, args_size=1
    0: iconst_0
    1: istore_1
    2: iload_1
    3: aload_0
    4: arraylength
    5: if_icmpge   23
    8: getstatic    #2           // Field java/
    ↳ lang/System.out:Ljava/io/PrintStream;
    11: aload_0
    12: iload_1
    13: iaload
    14: invokevirtual #3           // Method java/io/
    ↳ /PrintStream.println:(I)V
    17: iinc         1, 1
    20: goto        2

```

```
23: return
```

54.13.2

```
:  
  
public class ArraySum  
{  
    public static int f (int[] a)  
    {  
        int sum=0;  
        for (int i=0; i<a.length; i++)  
            sum=sum+a[i];  
        return sum;  
    }  
}
```

```
public static int f(int[]);  
flags: ACC_PUBLIC, ACC_STATIC  
Code:  
stack=3, locals=3, args_size=1  
 0:  iconst_0  
 1:  istore_1  
 2:  iconst_0  
 3:  istore_2  
 4:  iload_2  
 5:  aload_0  
 6:  arraylength  
 7:  if_icmpge   22  
10:  iload_1  
11:  aload_0  
12:  iload_2  
13:  iaload  
14:  iadd  
15:  istore_1  
16:  iinc          2, 1  
19:  goto          4  
22:  iload_1  
23:  ireturn
```

54.13.3

```
public class UseArgument
```

```
{
    public static void main(String[] args)
    {
        System.out.print("Hi, ");
        System.out.print(args[1]);
        System.out.println(". How are you?");
    }
}
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=3, locals=1, args_size=1
  0: getstatic      #2           // Field java/
  ↴ lang/System.out:Ljava/io/PrintStream;
    3: ldc            #3           // String Hi,
    5: invokevirtual #4           // Method java/io/
  ↴ /PrintStream.print:(Ljava/lang/String;)V
    8: getstatic      #2           // Field java/
  ↴ lang/System.out:Ljava/io/PrintStream;
    11: aload_0
    12: iconst_1
    13: aaload
    14: invokevirtual #4           // Method java/io/
  ↴ /PrintStream.print:(Ljava/lang/String;)V
    17: getstatic      #2           // Field java/
  ↴ lang/System.out:Ljava/io/PrintStream;
    20: ldc            #5           // String . How
  ↴ are you?
    22: invokevirtual #6           // Method java/io/
  ↴ /PrintStream.println:(Ljava/lang/String;)V
    25: return
```

54.13.4

```
class Month
{
    public static String[] months =
    {
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
```

```

        "August",
        "September",
        "October",
        "November",
        "December"
    };

    public String get_month (int i)
    {
        return months[i];
    }
}

```

```

public java.lang.String get_month(int);
flags: ACC_PUBLIC
Code:
stack=2, locals=2, args_size=2
  0: getstatic      #2                      // Field months:[L
↳ Ljava/lang/String;
  3: iload_1
  4: aaload
  5: areturn

```

```

static {};
flags: ACC_STATIC
Code:
stack=4, locals=0, args_size=0
  0: bipush         12
  2: anewarray      #3                      // class java/
↳ lang/String
  5: dup
  6: iconst_0
  7: ldc            #4                      // String January
  9: aastore
 10: dup
 11: iconst_1
 12: ldc            #5                      // String 
↳ February
 14: aastore
 15: dup
 16: iconst_2
 17: ldc            #6                      // String March
 19: aastore
 20: dup
 21: iconst_3
 22: ldc            #7                      // String April
 24: aastore

```

```

25: dup
26: iconst_4
27: ldc           #8                      // String May
29: aastore
30: dup
31: iconst_5
32: ldc           #9                      // String June
34: aastore
35: dup
36: bipush        6
38: ldc           #10                     // String July
40: aastore
41: dup
42: bipush        7
44: ldc           #11                     // String August
46: aastore
47: dup
48: bipush        8
50: ldc           #12                     // String ↵
↳ September
52: aastore
53: dup
54: bipush        9
56: ldc           #13                     // String October
58: aastore
59: dup
60: bipush        10
62: ldc           #14                     // String ↵
↳ November
64: aastore
65: dup
66: bipush        11
68: ldc           #15                     // String ↵
↳ December
70: aastore
71: putstatic     #2                      // Field months:[ ↵
↳ Ljava/lang/String;
74: return

```

54.13.5

```

public static void f(int... values)
{
    for (int i=0; i<values.length; i++)
        System.out.println(values[i]);
}

```

```
public static void main(String[] args)
{
    f (1,2,3,4,5);
}
```

```
public static void f(int...);
flags: ACC_PUBLIC, ACC_STATIC, ACC_VARARGS
Code:
stack=3, locals=2, args_size=1
 0:  iconst_0
 1:  istore_1
 2:  iload_1
 3:  aload_0
 4:  arraylength
 5:  if_icmpge     23
 8:  getstatic      #2           // Field java/
↳ lang/System.out:Ljava/io/PrintStream;
 11:  aload_0
 12:  iload_1
 13:  iaload
 14:  invokevirtual #3           // Method java/io/
↳ /PrintStream.println:(I)V
 17:  iinc          1, 1
 20:  goto          2
 23:  return
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=4, locals=1, args_size=1
 0:  iconst_5
 1:  newarray      int
 3:  dup
 4:  iconst_0
 5:  iconst_1
 6:  iastore
 7:  dup
 8:  iconst_1
 9:  iconst_2
10:  iastore
11:  dup
12:  iconst_2
13:  iconst_3
14:  iastore
15:  dup
16:  iconst_3
```

```

17:  iconst_4
18:  iastore
19:  dup
20:  iconst_4
21:  iconst_5
22:  iastore
23:  invokestatic #4           // Method f:([I)V
26:  return

```

```

    public PrintStream format(String format, Object... args)
    ↵ )

```

(<http://docs.oracle.com/javase/tutorial/java/data/numberformat.html>)

:

```

public static void main(String[] args)
{
    int i=123;
    double d=123.456;
    System.out.format("int: %d double: %f.%n", i, d);
}

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=7, locals=4, args_size=1
        0: bipush      123
        2: istore_1
        3: ldc2_w      #2           // double 123.456
    ↵ d
        6: dstore_2
        7: getstatic     #4           // Field java/
    ↵ lang/System.out:Ljava/io/PrintStream;
        10: ldc         #5           // String int: %d
    ↵ double: %f.%n
        12: iconst_2
        13: anewarray     #6           // class java/
    ↵ lang/Object
        16: dup
        17: iconst_0
        18: iload_1
        19: invokestatic #7           // Method java/
    ↵ lang/Integer.valueOf:(I)Ljava/lang/Integer;
        22: aastore

```

```

23: dup
24: iconst_1
25: dload_2
26: invokestatic #8           // Method java/lang/Double
   ↴ lang/Double.valueOf:(D)Ljava/lang/Double;
   29: aastore
   30: invokevirtual #9        // Method java/io/PrintStream
   ↴ /PrintStream.format:(Ljava/lang/String;[Ljava/lang/Object;)V
   ↴ ;)Ljava/io/PrintStream;
   33: pop
   34: return

```

54.13.6

:

```

public static void main(String[] args)
{
    int[][] a = new int[5][10];
    a[1][2]=3;
}

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=3, locals=2, args_size=1
0: iconst_5
1: bipush      10
3: multianewarray #2,  2           // class "[[I"
7: astore_1
8: aload_1
9: iconst_1
10: aaload
11: iconst_2
12: iconst_3
13: iastore
14: return

```

?

```

public static int get12 (int[][][] in)
{
    return in[1][2];
}

```

```
public static int get12(int[][][]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=2, locals=1, args_size=1
        0: aload_0
        1: iconst_1
        2: aaload
        3: iconst_2
        4: iaload
        5: ireturn
```

54.13.7

```
public static void main(String[] args)
{
    int[][][] a = new int[5][10][15];
    a[1][2][3]=4;
    get_elem(a);
}
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=3, locals=2, args_size=1
        0: iconst_5
        1: bipush      10
        3: bipush      15
        5: multianewarray #2,  3           // class "[][][I"
        9: astore_1
        10: aload_1
        11: iconst_1
        12: aaload
        13: iconst_2
        14: aaload
        15: iconst_3
        16: iconst_4
        17: iastore
        18: aload_1
        19: invokestatic #3           // Method ↵
    ↳ get_elem:([][][I)
        22: pop
        23: return
```

```
public static int get_elem (int[][][] a)
{
    return a[1][2][3];
}
```

```
public static int get_elem(int[][][]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
  0: aload_0
  1: iconst_1
  2: aaload
  3: iconst_2
  4: aaload
  5: iconst_3
  6: iaload
  7: ireturn
```

54.13.8

54.14

54.14.1

```
public static void main(String[] args)
{
    System.out.println("What is your name?");
    String input = System.console().readLine();
    System.out.println("Hello, "+input);
}
```

```
public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=3, locals=2, args_size=1
  0: getstatic      #2           // Field java/io/PrintStream;
  ↳ lang/System.out:Ljava/io/PrintStream;
    3: ldc            #3           // String What is?
  ↳ your name?
    5: invokevirtual #4           // Method java/io/PrintStream.println:(Ljava/lang/String;)V
  ↳ /PrintStream.println:(Ljava/lang/String;)V
    8: invokestatic   #5           // Method java/lang/System.console:()Ljava/io/Console;
```

```

11: invokevirtual #6                      // Method java/io/⤦
↳ /Console.readLine:()Ljava/lang/String;
14: astore_1
15: getstatic      #2                  // Field java/⤦
↳ lang/System.out:Ljava/io/PrintStream;
18: new           #7                  // class java/⤦
↳ lang/StringBuilder
21: dup
22: invokespecial #8                 // Method java/⤦
↳ lang/StringBuilder."<init>":()V
25: ldc            #9                  // String Hello,
27: invokevirtual #10                 // Method java/⤦
↳ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/⤦
↳ StringBuilder;
30: aload_1
31: invokevirtual #10                 // Method java/⤦
↳ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/⤦
↳ StringBuilder;
34: invokevirtual #11                 // Method java/⤦
↳ lang/StringBuilder.toString:()Ljava/lang/String;
37: invokevirtual #4                  // Method java/io/⤦
↳ /PrintStream.println:(Ljava/lang/String;)V
40: return

```

54.14.2

:

```

public class strings
{
    public static char test (String a)
    {
        return a.charAt(3);
    }

    public static String concat (String a, String b)
    {
        return a+b;
    }
}

```

```

public static char test(java.lang.String);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
0: aload_0

```

```

1:  iconst_3
2:  invokevirtual #2                                // Method java/
   ↴ lang/String.charAt:(I)C
3:  ireturn

```

```

public static java.lang.String concat(java.lang.String, java.lang.String);
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=2, locals=2, args_size=2
    0: new           #3                      // class java/
   ↴ lang/StringBuilder
      3: dup
      4: invokespecial #4                  // Method java/
   ↴ lang/StringBuilder."<init>":()V
      7: aload_0
      8: invokevirtual #5                  // Method java/
   ↴ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/String;
   ↴ StringBuilder;
      11: aload_1
      12: invokevirtual #5                  // Method java/
   ↴ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/String;
   ↴ StringBuilder;
      15: invokevirtual #6                  // Method java/
   ↴ lang/StringBuilder.toString():()Ljava/lang/String;
      18: areturn

```

:

```

public static void main(String[] args)
{
    String s="Hello!";
    int n=123;
    System.out.println("s=" + s + " n=" + n);
}

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
  stack=3, locals=3, args_size=1
    0: ldc           #2                      // String Hello!
    2: astore_1
    3: bipush        123
    5: istore_2
    6: getstatic     #3                      // Field java/
   ↴ lang/System.out:Ljava/io/PrintStream;

```

```

    9: new           #4                  // class java/
    ↳ lang/StringBuilder
      12: dup
      13: invokespecial #5             // Method java/
    ↳ lang/StringBuilder."<init>":()V
      16: ldc            #6                  // String s=
      18: invokevirtual #7             // Method java/
    ↳ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/
    ↳ StringBuilder;
      21: aload_1
      22: invokevirtual #7             // Method java/
    ↳ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/
    ↳ StringBuilder;
      25: ldc            #8                  // String n=
      27: invokevirtual #7             // Method java/
    ↳ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/
    ↳ StringBuilder;
      30: iload_2
      31: invokevirtual #9             // Method java/
    ↳ lang/StringBuilder.append:(I)Ljava/lang/StringBuilder;
      34: invokevirtual #10            // Method java/
    ↳ lang/StringBuilder.toString():()Ljava/lang/String;
      37: invokevirtual #11            // Method java/io/
    ↳ /PrintStream.println:(Ljava/lang/String;)V
      40: return

```

54.15

Listing 54.10: IncorrectMonthException.java

```

public class IncorrectMonthException extends Exception
{
    private int index;

    public IncorrectMonthException(int index)
    {
        this.index = index;
    }
    public int getIndex()
    {
        return index;
    }
}

```

Listing 54.11: Month2.java

```

class Month2
{
    public static String[] months =
    {
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
    };

    public static String get_month (int i) throws ↵
        IncorrectMonthException
    {
        if (i<0 || i>11)
            throw new IncorrectMonthException(i);
        return months[i];
    }

    public static void main (String[] args)
    {
        try
        {
            System.out.println(get_month(100));
        }
        catch(IncorrectMonthException e)
        {
            System.out.println("incorrect month ↵
                index: "+ e.getIndex());
            e.printStackTrace();
        }
    }
}

```

```

public IncorrectMonthException(int);
flags: ACC_PUBLIC
Code:
stack=2, locals=2, args_size=2
  0: aload_0
  1: invokespecial #1                         // Method java/

```

```

    ↳ lang/Exception."<init>":()V
        4: aload_0
        5: iload_1
        6: putfield      #2                  // Field index:I
        9: return

```

getIndex().

```

public int getIndex();
flags: ACC_PUBLIC
Code:
    stack=1, locals=1, args_size=1
    0: aload_0
    1: getfield      #2                  // Field index:I
    4: ireturn

```

Listing 54.12: Month2.class

```

public static java.lang.String get_month(int) throws ↳
    ↳ IncorrectMonthException;
flags: ACC_PUBLIC, ACC_STATIC
Code:
    stack=3, locals=1, args_size=1
    0: iload_0
    1: iflt          10
    4: iload_0
    5: bipush         11
    7: if_icmple     19
    10: new           #2                  // class ↳
    ↳ IncorrectMonthException
    13: dup
    14: iload_0
    15: invokespecial #3                  // Method ↳
    ↳ IncorrectMonthException."<init>":(I)V
    18: athrow
    19: getstatic      #4                  // Field months:[ ↳
    ↳ Ljava/lang/String;
    22: iload_0
    23: aaload
    24: areturn

```

? main() en Month2.class:

Listing 54.13: Month2.class

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:

```

```

stack=3, locals=2, args_size=1
  0: getstatic      #5           // Field java/
    ↴ lang/System.out:Ljava/io/PrintStream;
      3: bipush        100
      5: invokestatic   #6           // Method ↴
    ↴ get_month:(I)Ljava/lang/String;
      8: invokevirtual #7           // Method java/io/
    ↴ /PrintStream.println:(Ljava/lang/String;)V
      11: goto         47
      14: astore_1
      15: getstatic      #5           // Field java/
    ↴ lang/System.out:Ljava/io/PrintStream;
      18: new           #8           // class java/
    ↴ lang/StringBuilder
      21: dup
      22: invokespecial #9           // Method java/
    ↴ lang/StringBuilder."<init>":()V
      25: ldc           #10          // String ↴
    ↴ incorrect month index:
      27: invokevirtual #11          // Method java/
    ↴ lang/StringBuilder.append:(Ljava/lang/String;)Ljava/lang/
    ↴ StringBuilder;
      30: aload_1
      31: invokevirtual #12          // Method ↴
    ↴ IncorrectMonthException.getIndex:()I
      34: invokevirtual #13          // Method java/
    ↴ lang/StringBuilder.append:(I)Ljava/lang/StringBuilder;
      37: invokevirtual #14          // Method java/
    ↴ lang/StringBuilder.toString:()Ljava/lang/String;
      40: invokevirtual #7           // Method java/io/
    ↴ /PrintStream.println:(Ljava/lang/String;)V
      43: aload_1
      44: invokevirtual #15          // Method ↴
    ↴ IncorrectMonthException.printStackTrace:()V
      47: return
Exception table:
  from   to target type
    0     11    14   Class IncorrectMonthException

```

Listing 54.14:

```

.catch java/io/FileNotFoundException from met001_335 to ↴
  ↴ met001_360\
using met001_360
  .catch java/io/FileNotFoundException from met001_185 to ↴
  ↴ met001_214\
using met001_214

```

```
.catch java/io/FileNotFoundException from met001_181 to ↵
    ↴ met001_192\
using met001_195
    .catch java/io/FileNotFoundException from met001_155 to ↵
        ↴ met001_176\
using met001_176
    .catch java/io/FileNotFoundException from met001_83 to ↵
        ↴ met001_129 using \
met001_129
    .catch java/io/FileNotFoundException from met001_42 to ↵
        ↴ met001_66 using \
met001_69
    .catch java/io/FileNotFoundException from met001_begin to ↵
        ↴ met001_37\
using met001_37
```

54.16

:

Listing 54.15: test.java

```
public class test
{
    public static int a;
    private static int b;

    public test()
    {
        a=0;
        b=0;
    }
    public static void set_a (int input)
    {
        a=input;
    }
    public static int get_a ()
    {
        return a;
    }
    public static void set_b (int input)
    {
        b=input;
    }
    public static int get_b ()
    {
```

```
        return b;
    }
}
```

```
public test();
flags: ACC_PUBLIC
Code:
stack=1, locals=1, args_size=1
  0: aload_0
  1: invokespecial #1                      // Method java/lang/Object.<init>:()V
↳ lang/Object."<init>":()V
  4: iconst_0
  5: putstatic      #2                      // Field a:I
  8: iconst_0
  9: putstatic      #3                      // Field b:I
12: return
```

a:

```
public static void set_a(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
  0: iload_0
  1: putstatic      #2                      // Field a:I
  4: return
```

a:

```
public static int get_a();
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=0, args_size=0
  0: getstatic      #2                      // Field a:I
  3: ireturn
```

b:

```
public static void set_b(int);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=1, args_size=1
  0: iload_0
  1: putstatic      #3                      // Field b:I
  4: return
```

b:

```

public static int get_b();
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=1, locals=0, args_size=0
0: getstatic    #3           // Field b:I
3: ireturn

```

Listing 54.16: ex1.java

```

public class ex1
{
    public static void main(String[] args)
    {
        test obj=new test();
        obj.set_a (1234);
        System.out.println(obj.a);
    }
}

```

```

public static void main(java.lang.String[]);
flags: ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=2, args_size=1
0: new             #2           // class test
3: dup
4: invokespecial #3           // Method test."<init>":()V
    7: astore_1
    8: aload_1
    9: pop
    10: sipush      1234
    13: invokestatic #4           // Method test.set_a:(I)V
    16: getstatic    #5           // Field java/lang/System.out:Ljava/io/PrintStream;
    19: aload_1
    20: pop
    21: getstatic    #6           // Field test.a:I
    24: invokevirtual #7           // Method java/io/PrintStream.println:(I)V
    27: return

```

54.17

54.17.1

```

public class nag
{
    public static void nag_screen()
    {
        System.out.println("This program is not ↵
↳ registered");
    }
    public static void main(String[] args)
    {
        System.out.println("Greetings from the mega- ↵
↳ software");
        nag_screen();
    }
}

```

```

; Segment type: Pure code
.method public static nag_screen()V
.limit stack 2
.line 4
• 178 000 002 | getstatic java/lang/System.out Ljava/io/PrintStream; ; CODE XREF: main+84P
• 018 003     ldc "This program is not registered"
• 182 000 004     invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 5
• 177     return
• ??? ??? ???+ .end method
• ??? ??? ???+
???
;

; Segment type: Pure code
.method public static main([Ljava/lang/String;)V
.limit stack 2
.limit locals 1
.line 8
• 178 000 002 | getstatic java/lang/System.out Ljava/io/PrintStream;
• 018 005     ldc "Greetings from the mega-software"
• 182 000 004     invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 9
• 184 000 006     invokestatic nag.nag_screen()V
.line 10
• 177     return

```

Figura 54.1: IDA

```
; Segment type: Pure code
.method public static nag_screen()V
.limit stack 2
.line 4

nag_screen:                               ; CODE XREF: main+81P
    return
    0 : 0x00
    2 : 0x02
    ldc "This program is not registered"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
    .line 5
    return
.???.????????+.end method
.???.????????+
.???
```

Figura 54.2: IDA

(JRE 1.7):

```
Exception in thread "main" java.lang.VerifyError: Expecting a ↴
↳ stack map frame
Exception Details:
  Location:
    nag.nag_screen()V @1: nop
  Reason:
    Error exists in the bytecode
Bytecode:
  0000000: b100 0212 03b6 0004 b1

          at java.lang.Class.getDeclaredMethods0(Native Method)
          at java.lang.Class.privateGetDeclaredMethods(Class.java: ↴
↳ :2615)
          at java.lang.Class.getMethod0(Class.java:2856)
          at java.lang.Class.getMethod(Class.java:1668)
          at sun.launcher.LauncherHelper.getMainMethod( ↴
↳ LauncherHelper.java:494)
          at sun.launcher.LauncherHelper.checkAndLoadMain( ↴
↳ LauncherHelper.java:486)
```

```

; Segment type: Pure code
.method public static main([Ljava/lang/String;)V
.limit stack 2
.limit locals 1
.line 8
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "Greetings from the mega-software"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 9
    nop
    nop
    nop
.line 10
    return
.line 177
;
=====
```

Figura 54.3: IDA

!

54.17.2

```

public class password
{
    public static void main(String[] args)
    {
        System.out.println("Please enter the password");
        String input = System.console().readLine();
        if (input.equals("secret"))
            System.out.println("password is correct");
        else
            System.out.println("password is not correct");
    }
}
```

IDA:

```

; Segment type: Pure code
.method public static main([Ljava/lang/String;)V
.limit stack 2
.limit locals 2
.line 3
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "Please enter the password"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 4
    invokestatic java/lang/System.console()Ljava/io/Console;
    invokevirtual java/io/Console.readLine()Ljava/lang/String;
    astore_1 ; met002_slot001
.line 5
    aload_1 ; met002_slot001
    ldc "secret"
    invokevirtual java/lang/String.equals(Ljava/lang/Object;)Z
    ifeq met002_35
.line 6
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "password is correct"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
    goto met002_43
.line 8

met002_35:                                ; CODE XREF: main+21↑j
178 000 002
    .stack use locals
    .locals Object java/lang/String
    .end stack
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "password is not correct"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 9

```

Figura 54.4: IDA

```

; Segment type: Pure code
.method public static main([Ljava/lang/String;)V
.limit stack 2
.limit locals 2
.line 3
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "Please enter the password"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 4
    invokestatic java/lang/System.console()Ljava/io/Console;
    invokevirtual java/io/Console.readLine()Ljava/lang/String;
    astore_1 ; met002_slot001
.line 5
    aload_1 ; met002_slot001
    ldc "secret"
    invokevirtual java/lang/String.equals(Ljava/lang/Object;)Z
    ifeq met002_24
.line 6

met002_24:                                ; CODE XREF: main+21↑j
178 000 002
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "password is correct"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
    goto met002_43
.line 8
    .stack use locals
    .locals Object java/lang/String
    .end stack
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "password is not correct"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 9

```

Figura 54.5: IDA

```
Exception in thread "main" java.lang.VerifyError: Expecting a ↴
↳ stackmap frame at branch target 24
Exception Details:
  Location:
    password.main([Ljava/lang/String;)V @21: ifeq
  Reason:
    Expected stackmap frame at this location.
Bytecode:
  0000000: b200 0212 03b6 0004 b800 05b6 0006 4c2b
  0000010: 1207 b600 0899 0003 b200 0212 09b6 0004
  0000020: a700 0bb2 0002 120a b600 04b1
Stackmap Table:
  append_frame(@35, Object[#20])
  same_frame(@43)

  at java.lang.Class.getDeclaredMethods0(Native Method)
  at java.lang.Class.privateGetDeclaredMethods(Class.java: ↴
↳ :2615)
  at java.lang.Class.getMethod0(Class.java:2856)
  at java.lang.Class.getMethod(Class.java:1668)
  at sun.launcher.LauncherHelper.getMainMethod( ↴
↳ LauncherHelper.java:494)
  at sun.launcher.LauncherHelper.checkAndLoadMain( ↴
↳ LauncherHelper.java:486)
```

JRE 1.6.

```

; Segment type: Pure code
.method public static main([Ljava/lang/String;)V
.limit stack 2
.limit locals 2
.line 3
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "Please enter the password"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
.line 4
    invokestatic java/lang/System.console()Ljava/io/Console;
    invokevirtual java/io/Console.readline()Ljava/lang/String;
    astore_1 ; met002_slot001
.line 5
    iconst_1
    nop
    nop
    nop
    nop
    nop
    nop
    ifeq met002_35
.line 6
    getstatic java/lang/System.out Ljava/io/PrintStream;
    ldc "password is correct"
    invokevirtual java/io/PrintStream.println(Ljava/lang/String;)V
    goto met002_43
.line 8

met002_35:                                ; CODE XREF: main+21↑j
    .stack use locals
        locals Object java/lang/String
    .end stack

```

Figura 54.6: IDA

54.18

- .
- .
- .
- .

Parte V

<http://www.google.com/search?q=CsDecomprLZC>

Capítulo 55

55.1 Microsoft Visual C++

:

	CL.EXE			
6	6.0	12.00	msvcrt.dll, msycop60.dll	June 1998
.NET (2002)	7.0	13.00	msvcr70.dll, msycop70.dll	February 13, 2002
.NET 2003	7.1	13.10	msvcr71.dll, msycop71.dll	April 24, 2003
2005	8.0	14.00	msvcr80.dll, msycop80.dll	November 7, 2005
2008	9.0	15.00	msvcr90.dll, msycop90.dll	November 19, 2007
2010	10.0	16.00	msvcr100.dll, msycop100.dll	April 12, 2010
2012	11.0	17.00	msvcr110.dll, msycop110.dll	September 12, 2012
2013	12.0	18.00	msvcr120.dll, msycop120.dll	October 17, 2013

msycop*.dll .

55.1.1 Name mangling

?

[name mangling : 51.1.1 on page 702.](#)

55.2 GCC

Cygwin y MinGW.

55.2.1 Name mangling

_Z.

[name mangling : 51.1.1 on page 702.](#)

55.2.2 Cygwin

cygwin1.dll .

55.2.3 MinGW

msvcrt.dll .

55.3 Intel FORTRAN

libifcoremd.dll, libifportmd.dll y libiomp5md.dll (OpenMP) .

libifcoremd.dll for_, FORTRAN.

55.4 Watcom, OpenWatcom

55.4.1 Name mangling

W.

:

```
W?method$_class$n__v
```

55.5 Borland

:

```
@TApplication@IdleAction$qv
@TApplication@ProcessMDIAccels$qp6tagMSG
@TModule@$bctr$qpcpvt1
@TModule@$bdtr$qv
@TModule@ValidWindow$qp14TWindowsObject
@TrueColorTo8BitN$qpvi1iiit1iiiiii
@TrueColorTo16BitN$qpvi1iiit1iiiiii
@DIB24BitTo8BitBitmap$qpvi1iiit1iiiiii
@TrueBitmap@$bctr$qpcl
@TrueBitmap@$bctr$qpvl
@TrueBitmap@$bctr$qi1lll
```

@.

Spanish text placeholder.

Borland Visual Component Libraries (VCL) , vcl50.dll, rtl60.dll.

: BORLNDMM.DLL.

55.5.1 Delphi

«Boolean» .

:

00000400	04 10 40 00 03 07 42 6f	6f 6c 65 61 6e 01 00 00	... ↴
	↳ @...Boolean...		
00000410	00 00 01 00 00 00 00 10	40 00 05 46 61 6c 73 65	↗
	↳ @...False		
00000420	04 54 72 75 65 8d 40 00	2c 10 40 00 09 08 57 69	. ↴
	↳ True.@...@...Wi		
00000430	64 65 43 68 61 72 03 00	00 00 00 ff ff 00 00 90	↴
	↳ deChar.....		
00000440	44 10 40 00 02 04 43 68	61 72 01 00 00 00 00 ff	D. ↴
	↳ @...Char.....		
00000450	00 00 00 90 58 10 40 00	01 08 53 6d 61 6c 6c 69	↗
	↳ X.@...Smalli		
00000460	6e 74 02 00 80 ff ff ff	7f 00 00 90 70 10 40 00	nt ↴
	↳p.@.		
00000470	01 07 49 6e 74 65 67 65	72 04 00 00 00 80 ff ff	... ↴
	↳ Integer.....		
00000480	ff 7f 8b c0 88 10 40 00	01 04 42 79 74 65 01 00	↗
	↳ @...Byte..		
00000490	00 00 00 ff 00 00 00 90	9c 10 40 00 01 04 57 6f	↗
	↳ @...Wo		
000004a0	72 64 03 00 00 00 00 ff	ff 00 00 90 b0 10 40 00	rd ↴
	↳@.		
000004b0	01 08 43 61 72 64 69 6e	61 6c 05 00 00 00 00 ff	... ↴
	↳ Cardinal.....		
000004c0	ff ff ff 90 c8 10 40 00	10 05 49 6e 74 36 34 00	↗
	↳ @...Int64.		
000004d0	00 00 00 00 00 00 80 ff	ff ff ff ff ff ff 7f 90	↗
	↳ 		
000004e0	e4 10 40 00 04 08 45 78	74 65 6e 64 65 64 02 90	... ↴
	↳ @...Extended..		
000004f0	f4 10 40 00 04 06 44 6f	75 62 6c 65 01 8d 40 00	... ↴
	↳ @...Double..@.		
00000500	04 11 40 00 04 08 43 75	72 72 65 6e 63 79 04 90	... ↴
	↳ @...Currency..		

00000510	14 11 40 00 0a 06 73 74 72 69 6e 67 20 11 40 00 ...
	↳ @...string .@.
00000520	0b 0a 57 69 64 65 53 74 72 69 6e 67 30 11 40 00 ...
	↳ WideString0.@.
00000530	0c 07 56 61 72 69 61 6e 74 8d 40 00 40 11 40 00 ...
	↳ Variant.@@@.
00000540	0c 0a 4f 6c 65 56 61 72 69 61 6e 74 98 11 40 00 ...
	↳ OleVariant..@.
00000550	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ✓
	↳
00000560	00 00 00 00 00 00 00 00 00 00 00 00 98 11 40 00 ✓
	↳ @.
00000570	04 00 00 00 00 00 00 00 18 4d 40 00 24 4d 40 00 ✓
	↳ M@.\$M@.
00000580	28 4d 40 00 2c 4d 40 00 20 4d 40 00 68 4a 40 00 (✓
	↳ M@.,M@. M@.hJ@.
00000590	84 4a 40 00 c0 4a 40 00 07 54 4f 62 6a 65 63 74 .✓
	↳ J@..J@..TObject
000005a0	a4 11 40 00 07 07 54 4f 62 6a 65 63 74 98 11 40 ...
	↳ @...TObject..@
000005b0	00 00 00 00 00 00 00 06 53 79 73 74 65 6d 00 00 ✓
	↳ System..
000005c0	c4 11 40 00 0f 0a 49 49 6e 74 65 72 66 61 63 65 ...
	↳ @...IInterface
000005d0	00 00 00 00 01 00 00 00 00 00 00 00 c0 00 00 ✓
	↳
000005e0	00 00 00 00 46 06 53 79 73 74 65 6d 03 00 ff ff ✓
	↳ F.System....
000005f0	f4 11 40 00 0f 09 49 44 69 73 70 61 74 63 68 c0 ...
	↳ @...IDispatch.
00000600	11 40 00 01 00 04 02 00 00 00 00 00 c0 00 00 00 .>@
	↳
00000610	00 00 00 46 06 53 79 73 74 65 6d 04 00 ff ff 90 ✓
	↳ ...F.System....
00000620	cc 83 44 24 04 f8 e9 51 6c 00 00 83 44 24 04 f8 ...
	↳ D\$...Q1...D\$..
00000630	e9 6f 6c 00 00 83 44 24 04 f8 e9 79 6c 00 00 cc .✓
	↳ ol...D\$...y1...
00000640	cc 21 12 40 00 2b 12 40 00 35 12 40 00 01 00 00 ✓
	↳ !.@.+.@.5.@....
00000650	00 00 00 00 00 00 00 00 00 c0 00 00 00 00 00 00 ✓
	↳
00000660	46 41 12 40 00 08 00 00 00 00 00 00 00 00 8d 40 00 FA✓
	↳ .@.....@.
00000670	bc 12 40 00 4d 12 40 00 00 00 00 00 00 00 00 00 00 00 ...
	↳ @.M@.....
00000680	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ✓

```

↳ |.....| 00000690 bc 12 40 00 0c 00 00 00 4c 11 40 00 18 4d 40 00 |..✓
↳ @....L.@..M@.| 000006a0 50 7e 40 00 5c 7e 40 00 2c 4d 40 00 20 4d 40 00 |P~✓
↳ @..\~@.,M@. M@.| 000006b0 6c 7e 40 00 84 4a 40 00 c0 4a 40 00 11 54 49 6e |l~✓
↳ @..J@..J@..TIn| 000006c0 74 65 72 66 61 63 65 64 4f 62 6a 65 63 74 8b c0 |✓
↳ terfacedObject..| 000006d0 d4 12 40 00 07 11 54 49 6e 74 65 72 66 61 63 65 |..✓
↳ @...TInterface| 000006e0 64 4f 62 6a 65 63 74 bc 12 40 00 a0 11 40 00 00 |✓
↳ dObject..@...@..| 000006f0 00 06 53 79 73 74 65 6d 00 00 8b c0 00 13 40 00 |..✓
↳ System.....@.| 00000700 11 0b 54 42 6f 75 6e 64 41 72 72 61 79 04 00 00 |..✓
↳ TBoundArray...| 00000710 00 00 00 00 00 03 00 00 00 6c 10 40 00 06 53 79 ✓
↳ |.....1.@..Sy| 00000720 73 74 65 6d 28 13 40 00 04 09 54 44 61 74 65 54 |✓
↳ stem(@...TDateT| 00000730 69 6d 65 01 ff 25 48 e0 c4 00 8b c0 ff 25 44 e0 |✓
↳ ime.%H.....%D.|
```

00 00 00 00,32 13 8B C0 o FF FF FF FF.

55.6

- vcomp*.dll.

Capítulo 56

(win32)

: Process Monitor¹ SysInternals.

Wireshark².

: 75.

blog.yurichev.com.

56.1 Windows API

... CRT.

- (advapi32.dll): RegEnumKeyEx^{3 4}, RegEnumValue^{5 4}, RegGetValue^{6 4}, RegOpenKeyEx^{7 4}, RegQueryValueEx^{8 4}.
- (kernel32.dll): GetPrivateProfileString^{9 4}.

¹<http://go.yurichev.com/17301>

²<http://go.yurichev.com/17303>

³MSDN

⁴

⁵MSDN

⁶MSDN

⁷MSDN

⁸MSDN

⁹MSDN

- (user32.dll): MessageBox ¹⁰ ⁴, MessageBoxEx ¹¹ ⁴, SetDlgItemText ¹² ⁴, GetDlgItemText ¹³ ⁴.
- ([68.2.8 on page 906](#)) : (user32.dll): LoadMenu ¹⁴ ⁴.
- (ws2_32.dll): WSARecv ¹⁵, WSASend ¹⁶.
- (kernel32.dll): CreateFile ¹⁷ ⁴, ReadFile ¹⁸, ReadFileEx ¹⁹, WriteFile ²⁰, WriteFileEx ²¹.
- Internet (wininet.dll): WinHttpOpen ²².
- (wintrust.dll): WinVerifyTrust ²³.
- (msvcr*.dll): assert, itoa, ltoa, open, printf, read, strcmp, atol, atoi, fopen, fread, fwrite, memcmp, rand, strlen, strstr, strchr.

56.2 tracer:

```
--one-time-INT3-bp:somedll.dll!.*
```

```
--one-time-INT3-bp:somedll.dll!xml.*
```

Y hay muchas funciones.

```
tracer -l:uptime.exe --one-time-INT3-bp:cygwin1.dll!.*
```

:

```
One-time INT3 breakpoint: cygwin1.dll!__main (called from ↴
 ↴ uptime.exe!OEP+0x6d (0x40106d))
```

¹⁰[MSDN](#)

¹¹[MSDN](#)

¹²[MSDN](#)

¹³[MSDN](#)

¹⁴[MSDN](#)

¹⁵[MSDN](#)

¹⁶[MSDN](#)

¹⁷[MSDN](#)

¹⁸[MSDN](#)

¹⁹[MSDN](#)

²⁰[MSDN](#)

²¹[MSDN](#)

²²[MSDN](#)

²³[MSDN](#)

```
One-time INT3 breakpoint: cygwin1.dll!_geteuid32 (called from ↵
    ↳ uptime.exe!0EP+0xba3 (0x401ba3))
One-time INT3 breakpoint: cygwin1.dll!_getuid32 (called from ↵
    ↳ uptime.exe!0EP+0xbba (0x401baa))
One-time INT3 breakpoint: cygwin1.dll!_getegid32 (called from ↵
    ↳ uptime.exe!0EP+0xcb7 (0x401cb7))
One-time INT3 breakpoint: cygwin1.dll!_getgid32 (called from ↵
    ↳ uptime.exe!0EP+0xcbe (0x401cbe))
One-time INT3 breakpoint: cygwin1.dll!sysconf (called from ↵
    ↳ uptime.exe!0EP+0x735 (0x401735))
One-time INT3 breakpoint: cygwin1.dll!setlocale (called from ↵
    ↳ uptime.exe!0EP+0x7b2 (0x4017b2))
One-time INT3 breakpoint: cygwin1.dll!_open64 (called from ↵
    ↳ uptime.exe!0EP+0x994 (0x401994))
One-time INT3 breakpoint: cygwin1.dll!_lseek64 (called from ↵
    ↳ uptime.exe!0EP+0x7ea (0x4017ea))
One-time INT3 breakpoint: cygwin1.dll!read (called from uptime.↗
    ↳ exe!0EP+0x809 (0x401809))
One-time INT3 breakpoint: cygwin1.dll!sscanf (called from ↵
    ↳ uptime.exe!0EP+0x839 (0x401839))
One-time INT3 breakpoint: cygwin1.dll!uname (called from uptime.↗
    ↳ .exe!0EP+0x139 (0x401139))
One-time INT3 breakpoint: cygwin1.dll!time (called from uptime.↗
    ↳ exe!0EP+0x22e (0x40122e))
One-time INT3 breakpoint: cygwin1.dll!localtime (called from ↵
    ↳ uptime.exe!0EP+0x236 (0x401236))
One-time INT3 breakpoint: cygwin1.dll!sprintf (called from ↵
    ↳ uptime.exe!0EP+0x25a (0x40125a))
One-time INT3 breakpoint: cygwin1.dll!setutent (called from ↵
    ↳ uptime.exe!0EP+0x3b1 (0x4013b1))
One-time INT3 breakpoint: cygwin1.dll!getutent (called from ↵
    ↳ uptime.exe!0EP+0x3c5 (0x4013c5))
One-time INT3 breakpoint: cygwin1.dll!endutent (called from ↵
    ↳ uptime.exe!0EP+0x3e6 (0x4013e6))
One-time INT3 breakpoint: cygwin1.dll!puts (called from uptime.↗
    ↳ exe!0EP+0x4c3 (0x4014c3))
```

Capítulo 57

57.1

57.1.1 C/C++

(ASCII).

. [Rit79] :

A minor difference was that the unit of I/O was the word, not the byte, because the PDP-7 was a word-addressed machine. In practice this meant merely that all programs dealing with character streams ignored null characters, because null was used to pad a file to an even number of characters.

:

```
int main()
{
    printf ("Hello, world!\n");
}
```

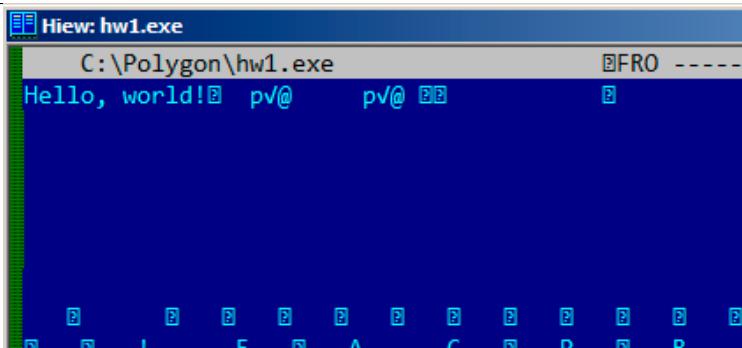


Figura 57.1: Hiew

57.1.2 Borland Delphi

:

Listing 57.1: Delphi

```

CODE:00518AC8          dd 19h
CODE:00518ACC aLoading__Plea db 'Loading... , please wait.',0
...
CODE:00518AFC          dd 10h
CODE:00518B00 aPreparingRun__ db 'Preparing run...',0

```

57.1.3 Unicode

...

: UTF-8 () y UTF-16LE (Windows).

UTF-8

UTF-8 ...

1:

¹: <http://go.yurichev.com/17304>

How much? 100€?

(English) I can eat glass and it doesn't hurt me.

(Greek) Μπορώ να φάω σπασμένα γυαλιά χωρίς να πάθω τίποτα.

(Hungarian) Meg tudom enni az üveget, nem lesz tőle bajom.

(Icelandic) Ég get etið gler án þess að meiða mig.

(Polish) Mogę jeść szkło i mi nie szkodzi.

(Russian) Я могу есть стекло, оно мне не вредит.

(Arabic) أنا قادر على أكل الزجاج و هذا لا يُؤلمني.

(Hebrew) אֵנִי יָכוֹל לְאֶכְלָה זַהֲבָה וְזַהֲבָה לֹא מַזְרָעָה.

(Chinese) 我能吞下玻璃而不伤身体。

(Japanese) 私はガラスを食べられます。それは私を傷つけません。

(Hindi) मैं काँच खा सकता हूँ और मुझे उससे कोई चोट नहीं पहुंचती.

```
view hw4_UTF8.txt - Far 3.0.4040 x64 Administrator
hw4_UTF8.txt t 437 872 col 0 100% 0
How much? 100€?

(English) I can eat glass and it doesn't hurt me.
(Greek) Μπορώ να φάω σπασμένα γυαλιά χωρίς να πάθω τίποτα.
(Hungarian) Meg tudom enni az üveget, nem lesz tőle bajom.
(Icelandic) Ég get etið gler án þess að meiða mig.
(Polish) Mogę jeść szkło i mi nie szkodzi.
(Russian) Я могу есть стекло, оно мне не вредит.
(Arabic) أنا قادر على أكل الزجاج و هذا لا يُؤلمني.
(Hebrew) אֵנִי יָכוֹל לְאֶכְלָה זַהֲבָה וְזַהֲבָה לֹא מַזְרָעָה.
(Chinese) 我能吞下玻璃而不伤身体。
(Japanese) 私はガラスを食べられます。それは私を傷つけません。
(Hindi) मैं काँच खा सकता हूँ और मुझे उससे कोई चोट नहीं पहुंचती.
```

Figura 57.2: FAR: UTF-8

.

. BOM².

UTF-16LE

-A y -W. (wide). .

:

```
int wmain()
{
    wprintf(L"Hello, world!\n");
}
```

²Byte order mark

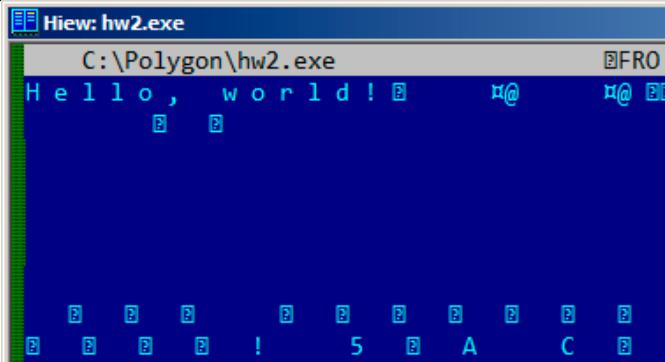


Figura 57.3: Hiew

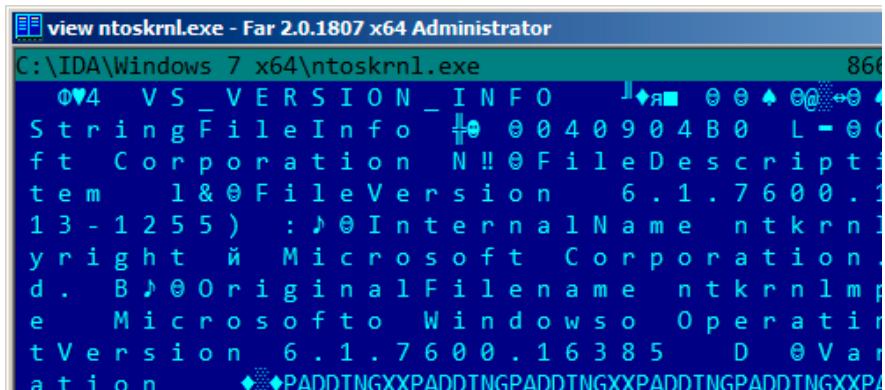


Figura 57.4: Hiew

```
.data:0040E000 aHelloWorld:  
.data:0040E000          unicode 0, <Hello, world!>  
.data:0040E000          dw 0Ah, 0
```

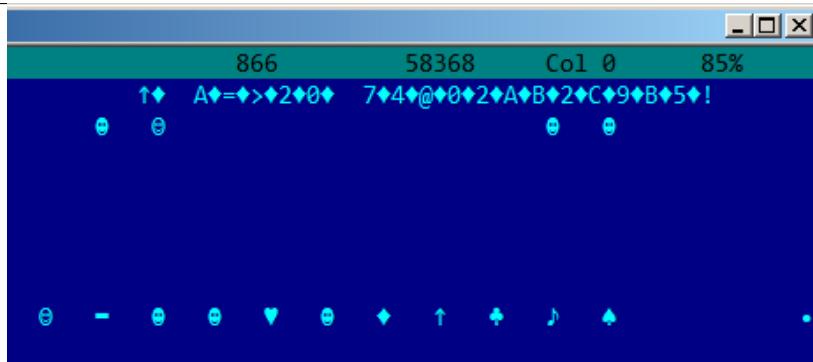


Figura 57.5: Hiew: UTF-16LE

.³ 0x400-0x4FF.

..

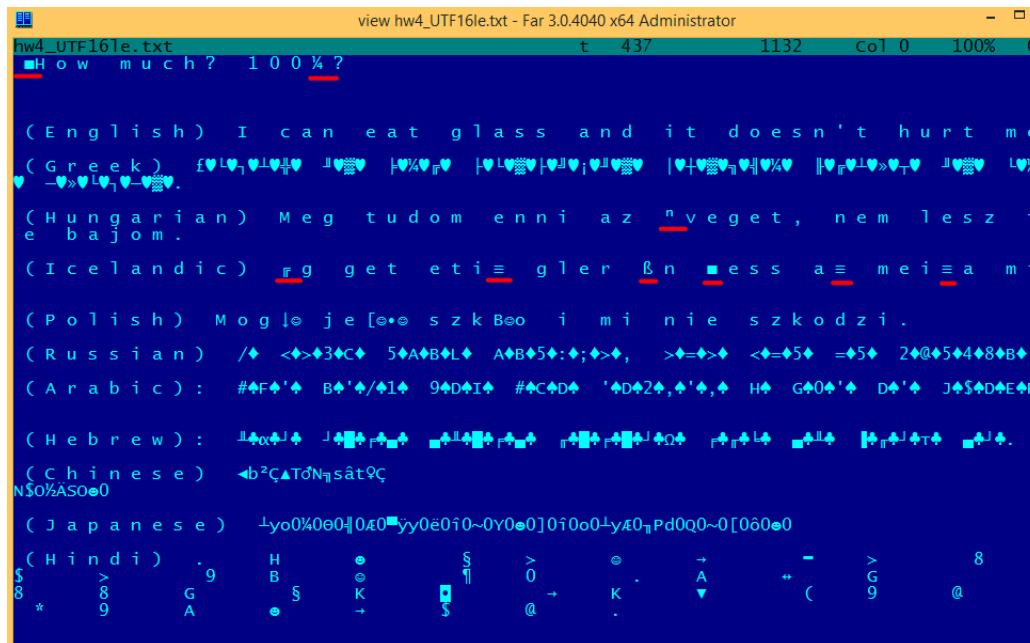


Figura 57.6: FAR: UTF-16LE

³wikipedia

57.1.4 Base64

AVjbbVSVfcUMu1xvj aMgjNtueRwBbxnyJw8dpGnLW8ZW8aKG3v4Y0icuQT+ ↴
↳ qEJAp91AOuWs=

WjbbVSVfcUMu1xvj aMgjNtueRwBbxnyJw8dpGnLW8ZW8aKG3v4Y0icuQT+ ↴
↳ qEJAp91AOuQ==

57.2

⁴.

: blog.yurichev.com.

: 78.2 on page 986.

57.3

⁵.

http://192.168.0.1/userRpmNatDebugRpm26525557/start_art.html.

base64.

⁴blog.yurichev.com

⁵<http://sekurak.pl/tp-link-httptftp-backdoor/>

Capítulo 58

Listing 58.1:

```
.text:107D4B29 mov  dx, [ecx+42h]
.text:107D4B2D cmp  edx, 1
.text:107D4B30 jz   short loc_107D4B4A
.text:107D4B32 push 1ECh
.text:107D4B37 push offset aWrite_c ; "write.c"
.text:107D4B3C push offset aTdTd_planarcon ; "td->Td->Td_planarconfig == PLANARCONFIG_CON"...
.text:107D4B41 call ds:_assert

...
.text:107D52CA mov  edx, [ebp-4]
.text:107D52CD and  edx, 3
.text:107D52D0 test edx, edx
.text:107D52D2 jz   short loc_107D52E9
.text:107D52D4 push 58h
.text:107D52D6 push offset aDumpmode_c ; "dumpmode.c"
.text:107D52DB push offset aN30      ; "(n & 3) == 0"
.text:107D52E0 call ds:_assert

...
.text:107D6759 mov  cx, [eax+6]
.text:107D675D cmp  ecx, 0Ch
.text:107D6760 jle  short loc_107D677A
.text:107D6762 push 2D8h
.text:107D6767 push offset aLzw_c  ; "lzw.c"
.text:107D676C push offset aSpLzw_nbBitsBit ; "sp->lzw_nbBits <= BITS_MAX"
.text:107D6771 call ds:_assert
```

Capítulo 59

10, 100, 1000, .

: 10=0xA, 100=0x64, 1000=0x3E8, 10000=0x2710.

0xAAAAAAAA (10101010101010101010101010101010) y
0x55555555 (01010101010101010101010101010101). 0x55AA , MBR¹, y en
ROM².

```
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476
```

Si encuentras estas cuatro constantes usadas en el código en fila, es muy altamente probable que esta función esté relacionada con MD5.

:

Listing 59.1: linux/lib/crc16.c

```
/** CRC table for the CRC-16. The poly is 0x8005 (x^16 + x^15 + x^2 + 1) */
u16 const crc16_table[256] = {
    0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0x0000,
    0xC241, 0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0000,
    0x0440, 0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0000,
    0x00E40,
    ...
}
```

: 37 on page 607.

¹Master Boot Record

²Memoria de solo lectura

59.1 Magic numbers

«MZ»³.

```
cmp [buf], 0x6468544D ; "MThd"
jnz _error_not_a_MIDI_file
```

59.1.1 DHCP

DhcpExtractOptionsForValidation() y DhcpExtractFullOptions():

Listing 59.2: dhcpcore.dll (Windows 7 x64)

```
.rdata:000007FF6483CBE8 dword_7FF6483CBE8 dd 63538263h
    ↴ ; DATA XREF: DhcpExtractOptionsForValidation+79
.rdata:000007FF6483CBEC dword_7FF6483CBEC dd 63538263h
    ↴ ; DATA XREF: DhcpExtractFullOptions+97
```

Listing 59.3: dhcpcore.dll (Windows 7 x64)

```
.text:000007FF6480875F mov     eax, [rsi]
.text:000007FF64808761 cmp     eax, cs:dword_7FF6483CBE8
.text:000007FF64808767 jnz     loc_7FF64817179
```

Listing 59.4: dhcpcore.dll (Windows 7 x64)

```
.text:000007FF648082C7 mov     eax, [r12]
.text:000007FF648082CB cmp     eax, cs:dword_7FF6483CBEC
.text:000007FF648082D1 jnz     loc_7FF648173AF
```

59.2

*binary grep*⁴.

³wikipedia

⁴GitHub

Capítulo 60

se puede intentar verificar cada una manualmente con un depurador.

operando obviamente, no nos sirven):

```
cat EXCEL.lst | grep fdiv | grep -v dbl_ > EXCEL.fdiv
```

...entonces vemos que hay 144 de ellas.

Spanish text placeholder

```
.text:3011E919 DC 33          fdiv      ↴
    ↳ qword ptr [ebx]
```

```
PID=13944|TID=28744|(0) 0x2f64e919 (Excel.exe!BASE+0x11e919)
EAX=0x02088006 EBX=0x02088018 ECX=0x00000001 EDX=0x00000001
ESI=0x02088000 EDI=0x00544804 EBP=0x0274FA3C ESP=0x0274F9F8
EIP=0x2F64E919
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=
FPU ST(0): 1.000000
```

[EBX].

```
.text:3011E91B DD 1E          fstp    ↵
    ↴ qword ptr [esi]
```

```
PID=32852|TID=36488|(0) 0x2f40e91b (Excel.exe!BASE+0x11e91b)
EAX=0x00598006 EBX=0x00598018 ECX=0x00000001 EDX=0x00000001
ESI=0x00598000 EDI=0x00294804 EBP=0x026CF93C ESP=0x026CF8F8
EIP=0x2F40E91B
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=C1 P
FPU ST(0): 0.333333
```

También como broma práctica, podemos modificarlo sobre la marcha:

```
tracer -l:excel.exe bpx=excel.exe!BASE+0x11E91B, set(st0,666)
```

```
PID=36540|TID=24056|(0) 0x2f40e91b (Excel.exe!BASE+0x11e91b)
EAX=0x00680006 EBX=0x00680018 ECX=0x00000001 EDX=0x00000001
ESI=0x00680000 EDI=0x00395404 EBP=0x0290FD9C ESP=0x0290FD58
EIP=0x2F40E91B
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=C1 P
FPU ST(0): 0.333333
Set ST0 register to 666.000000
```

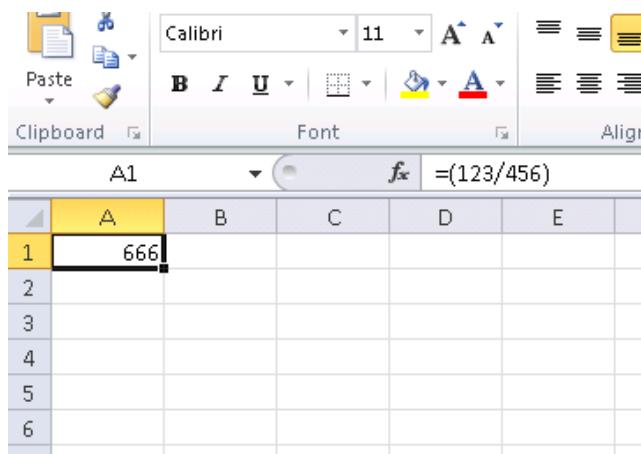


Figura 60.1:

```
tracer.exe -l:excel.exe bpx=excel.exe!BASE+0x1B7FCC, set(st0 ↵
↳ ,666)
```

Capítulo 61

61.1

XOR op, op (, XOR EAX, EAX) «». . Spanish text placeholder.

:

```
gawk -e '$2=="xor" { tmp=substr($3, 0, length($3)-1); if (tmp!=  
↳ $4) if($4!="esp") if ($4!="ebp") { print $1, $2, tmp,  
↳ ", ", $4 } }' filename.lst
```

([49 on page 688](#)).

61.2

Los compiladores modernos no emiten las instrucciones LOOP y RCL. Por otro lado, estas instrucciones son bien conocidas por los codificadores que les gusta programar directamente en lenguaje ensamblador. Si las ves, se puede decir que hay una alta probabilidad de que este fragmento de código haya sido escrito a mano. Tales instrucciones están marcadas como (M) en la lista de instrucciones en este apéndice: [A.6 on page 1230](#).

Windows 2003 (ntoskrnl.exe):

```

MultiplyTest proc near ; CODE XREF: ↵
    ↳ Get386Stepping
        xor    cx, cx
loc_620555:           ; CODE XREF: MultiplyTest+↗
    ↳ E
        push   cx
        call    Multiply
        pop    cx
        jb     short locret_620563
        loop   loc_620555
        clc
locret_620563:         ; CODE XREF: MultiplyTest+↗
    ↳ C
        retn
MultiplyTest endp

Multiply      proc near ; CODE XREF: MultiplyTest↗
    ↳ +5
        mov    ecx, 81h
        mov    eax, 417A000h
        mul    ecx
        cmp    edx, 2
        stc
        jnz    short locret_62057F
        cmp    eax, 0FE7A000h
        stc
        jnz    short locret_62057F
        clc
locret_62057F:          ; CODE XREF: Multiply+10
                           ; Multiply+18
        retn
Multiply      endp

```

[WRK¹](#) v1.2 WRK-v1.2\base\ntos\ke\i386\cpu.asm.

¹Windows Research Kernel

Capítulo 62

:	
0x150bf66 (_kziaia+0x14), e=	1 [MOV EBX, [EBP+8]] [EBP ↴
↳ +8]=0xf59c934	
0x150bf69 (_kziaia+0x17), e=	1 [MOV EDX, [69AEB08h]] [69 ↴
↳ AEB08h]=0	
0x150bf6f (_kziaia+0x1d), e=	1 [FS: MOV EAX, [2Ch]]
0x150bf75 (_kziaia+0x23), e=	1 [MOV ECX, [EAX+EDX*4]] [↴
↳ EAX+EDX*4]=0xf1ac360	
0x150bf78 (_kziaia+0x26), e=	1 [MOV [EBP-4], ECX] ECX=0 ↴
↳ xf1ac360	

Capítulo 63

63.1

63.2 C++

[RTTI¹ \(51.1.5 on page 721\)](#)-.

63.3

:

¹Run-time type information

Hiew: FW96650A.bin

		FRO			
00005000:	A0 B0 02 3C	-04 00 BE AF	-40 00 43 8C	-21 F0 A0 03	a@<@_!@
00005010:	FF 1F 02 3C	-21 E8 C0 03	-FF FF 42 34	-24 10 62 00	@<!шL@
00005020:	00 A0 03 3C	-25 10 43 00	-04 00 BE 8F	-08 00 E0 03	a@<%@C @
00005030:	08 00 BD 27	-F8 FF BD 27	-A0 B0 02 3C	-04 00 BE AF	шJ..@_!@
00005040:	48 00 43 8C	-21 F0 A0 03	-FF 1F 02 3C	-21 E8 C0 03	Н CM!Ёа@ В
00005050:	FF FF 42 34	-24 10 62 00	-00 A0 03 3C	-25 10 43 00	B4\$@b a
00005060:	04 00 BE 8F	-08 00 E0 03	-08 00 BD 27	-F8 FF BD 27	шJ@П@ р@@@
00005070:	21 10 00 00	-04 00 BE AF	-08 00 80 14	-21 F0 A0 03	!@ @ шJ@П@
00005080:	A0 B0 03 3C	-21 E8 C0 03	-44 29 02 7C	-3C 00 62 AC	a@<!шL@D)
00005090:	04 00 BE 8F	-08 00 E0 03	-08 00 BD 27	-01 00 03 24	шJ@П@ р@@@
000050A0:	44 29 62 7C	-A0 B0 03 3C	-21 E8 C0 03	-3C 00 62 AC	D)@ a@<!шL@
000050B0:	04 00 BE 8F	-08 00 E0 03	-08 00 BD 27	-F8 FF BD 27	шJ@П@ р@@@
000050C0:	A0 B0 02 3C	-04 00 BE AF	-84 00 43 8C	-21 F0 A0 03	a@<@_!@пД
000050D0:	21 E8 C0 03	-C4 FF 03 7C	-84 00 43 AC	-04 00 BE 8F	!шL@-ш Д
000050E0:	08 00 E0 03	-08 00 BD 27	-F8 FF BD 27	-A0 B0 02 3C	шJ@П@ J..@
000050F0:	04 00 BE AF	-20 00 43 8C	-21 F0 A0 03	-01 00 04 24	шJ@П CM!Ё
00005100:	21 E8 C0 03	-44 08 83 7C	-20 00 43 AC	-04 00 BE 8F	!шL@D@Г
00005110:	08 00 E0 03	-08 00 BD 27	-F8 FF BD 27	-A0 B0 02 3C	шJ@П@ J..@
00005120:	04 00 BE AF	-20 00 43 8C	-21 F0 A0 03	-21 E8 C0 03	шJ@П CM!Ё
00005130:	44 08 03 7C	-20 00 43 AC	-04 00 BE 8F	-08 00 E0 03	D@ CM@
00005140:	08 00 BD 27	-F8 FF BD 27	-A0 B0 03 3C	-04 00 BE AF	шJ..@_!@
00005150:	10 00 62 8C	-01 00 08 24	-04 A5 02 7D	-08 00 09 24	шJ@П \$@е@
00005160:	10 00 62 AC	-04 7B 22 7D	-04 48 02 7C	-04 84 02 7D	шJ@П {"}@Н
00005170:	10 00 62 AC	-21 F0 A0 03	-21 18 00 00	-A0 B0 0B 3C	шJ@П CM!Ёа@!@
00005180:	51 00 0A 24	-02 00 88 94	-00 00 89 94	-00 44 08 00	Q \$@ ИФ
00005190:	25 40 09 01	-01 00 63 24	-14 00 68 AD	-F9 FF 6A 14	%@@@@ c\$@
000051A0:	04 00 84 24	-21 18 00 00	-A0 B0 0A 3C	-07 00 09 24	шJ@П D\$!@ а
000051B0:	02 00 A4 94	-00 00 A8 94	-00 24 04 00	-25 20 88 00	шJ@П дФ ИФ \$

1Global 2FilBlk 3CryBlk 4ReLoad 5 6String 7Direct 8Table 9

Figura 63.1: Hiew:

: 86 on page 1123.

63.4

wikipedia.

: 85 on page 1116.

63.4.2

Parte VI

Capítulo 64

64.1 cdecl

Listing 64.1: cdecl

```
push arg3
push arg2
push arg1
call function
add esp, 12 ; returns ESP
```

64.2 stdcall

Listing 64.2: stdcall

```
push arg3
push arg2
push arg1
call function

function:
... do something ...
ret 12
```

Este método es ubicuo en las bibliotecas estándar de win32, pero no en win64 (ver más abajo sobre win64).

[8.1 on page 114](#) __stdcall:

```
int __stdcall f2 (int a, int b, int c)
{
    return a*b+c;
```

```
};
```

[8.2 on page 114](#), RET 12 RET. SP .

RETN n .

Listing 64.3: MSVC 2010

```
_a$ = 8 ; size ↗
↳ = 4
_b$ = 12 ; size ↗
↳ = 4
_c$ = 16 ; size ↗
↳ = 4
_f2@12 PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    imul    eax, DWORD PTR _b$[ebp]
    add     eax, DWORD PTR _c$[ebp]
    pop     ebp
    ret     12 ; ↗
    ↳ 0000000cH
_f2@12 ENDP

; ...
    push    3
    push    2
    push    1
    call    _f2@12
    push    eax
    push    OFFSET $SG81369
    call    _printf
    add     esp, 8
```

64.2.1

64.3 fastcall

Listing 64.4: fastcall

```
push arg3
mov edx, arg2
mov ecx, arg1
call function

function:
```

```
.. do something ..
ret 4
```

[8.1 on page 114](#) __fastcall:

```
int __fastcall f3 ( int a, int b, int c )
{
    return a*b+c;
};
```

:

Listing 64.5: Con optimización MSVC 2010 /Ob0

```
_c$ = 8                                ; size 2
    ↓ = 4
@f3@12 PROC
; _a$ = ecx
; _b$ = edx
    mov    eax, ecx
    imul   eax, edx
    add    eax, DWORD PTR _c$[esp-4]
    ret    4
@f3@12 ENDP

; ...

    mov    edx, 2
    push   3
    lea    ecx, DWORD PTR [edx-1]
    call   @f3@12
    push   eax
    push   OFFSET $SG81390
    call   _printf
    add    esp, 8
```

SP RETN .

64.3.1 GCC regparm

([19.1.1 on page 391](#)).

64.3.2 Watcom/OpenWatcom

«register calling convention». EAX, EDX, EBX y ECX... .

64.4 thiscall

(51.1.1 on page 702).

64.5 x86-64

64.5.1 Windows x64

```
.
:
#include <stdio.h>

void f1(int a, int b, int c, int d, int e, int f, int g)
{
    printf ("%d %d %d %d %d %d\n", a, b, c, d, e, f, g);
};

int main()
{
    f1(1,2,3,4,5,6,7);
}
```

Listing 64.6: MSVC 2012 /O**b**

```
$SG2937 DB      '%d %d %d %d %d %d %d', 0aH, 00H

main    PROC
        sub     rsp, 72
        ↳ 00000048H ; ↴

        mov     DWORD PTR [rsp+48], 7
        mov     DWORD PTR [rsp+40], 6
        mov     DWORD PTR [rsp+32], 5
        mov     r9d, 4
        mov     r8d, 3
        mov     edx, 2
        mov     ecx, 1
        call    f1

        xor     eax, eax
        add     rsp, 72
        ↳ 00000048H ; ↴
        ret    0
main    ENDP
```

```

a$ = 80
b$ = 88
c$ = 96
d$ = 104
e$ = 112
f$ = 120
g$ = 128
f1      PROC
$LN3:
    mov     DWORD PTR [rsp+32], r9d
    mov     DWORD PTR [rsp+24], r8d
    mov     DWORD PTR [rsp+16], edx
    mov     DWORD PTR [rsp+8], ecx
    sub    rsp, 72 ; ↴
    ↳ 00000048H

    mov     eax, DWORD PTR g$[rsp]
    mov     DWORD PTR [rsp+56], eax
    mov     eax, DWORD PTR f$[rsp]
    mov     DWORD PTR [rsp+48], eax
    mov     eax, DWORD PTR e$[rsp]
    mov     DWORD PTR [rsp+40], eax
    mov     eax, DWORD PTR d$[rsp]
    mov     DWORD PTR [rsp+32], eax
    mov     r9d, DWORD PTR c$[rsp]
    mov     r8d, DWORD PTR b$[rsp]
    mov     edx, DWORD PTR a$[rsp]
    lea    rcx, OFFSET FLAT:$SG2937
    call   printf

    add    rsp, 72 ; ↴
    ↳ 00000048H
    ret    0
f1      ENDP

```

Listing 64.7: Con optimización MSVC 2012 /Ob

\$SG2777 DB	'%d %d %d %d %d %d %d', 0aH, 00H
a\$ = 80	
b\$ = 88	
c\$ = 96	
d\$ = 104	
e\$ = 112	
f\$ = 120	
g\$ = 128	

```

f1      PROC
$LN3:
    sub     rsp, 72 ; ↴
    ↳ 00000048H

        mov     eax, DWORD PTR g$[rsp]
        mov     DWORD PTR [rsp+56], eax
        mov     eax, DWORD PTR f$[rsp]
        mov     DWORD PTR [rsp+48], eax
        mov     eax, DWORD PTR e$[rsp]
        mov     DWORD PTR [rsp+40], eax
        mov     DWORD PTR [rsp+32], r9d
        mov     r9d, r8d
        mov     r8d, edx
        mov     edx, ecx
        lea     rcx, OFFSET FLAT:$SG2777
        call    printf

        add     rsp, 72 ; ↴
    ↳ 00000048H
        ret     0
f1      ENDP

main   PROC
    sub     rsp, 72 ; ↴
    ↳ 00000048H

        mov     edx, 2
        mov     DWORD PTR [rsp+48], 7
        mov     DWORD PTR [rsp+40], 6
        lea     r9d, QWORD PTR [rdx+2]
        lea     r8d, QWORD PTR [rdx+1]
        lea     ecx, QWORD PTR [rdx-1]
        mov     DWORD PTR [rsp+32], 5
        call    f1

        xor     eax, eax
        add     rsp, 72 ; ↴
    ↳ 00000048H
        ret     0
main   ENDP

```

..
[:74.1 on page 952.](#)

Windows x64: (C/C++)

this RCX, RDX, Spanish text placeholder. : [51.1.1 on page 705](#).

64.5.2 Linux x64

Listing 64.8: Con optimización GCC 4.7.3

```
.LC0:
    .string "%d %d %d %d %d %d\n"
f1:
    sub    rsp, 40
    mov    eax, DWORD PTR [rsp+48]
    mov    DWORD PTR [rsp+8], r9d
    mov    r9d, ecx
    mov    DWORD PTR [rsp], r8d
    mov    ecx, esi
    mov    r8d, edx
    mov    esi, OFFSET FLAT:.LC0
    mov    edx, edi
    mov    edi, 1
    mov    DWORD PTR [rsp+16], eax
    xor    eax, eax
    call   __printf_chk
    add    rsp, 40
    ret
main:
    sub    rsp, 24
    mov    r9d, 6
    mov    r8d, 5
    mov    DWORD PTR [rsp], 7
    mov    ecx, 4
    mov    edx, 3
    mov    esi, 2
    mov    edi, 1
    call   f1
    add    rsp, 24
    ret
```

64.6**64.7**

```
#include <stdio.h>
```

```
void f(int a, int b)
{
    a=a+b;
    printf ("%d\n", a);
}
```

Listing 64.9: MSVC 2012

```
_a$ = 8                                ; size 4
    ↓ = 4
_b$ = 12                               ; size 4
    ↓ = 4
_f PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    add     eax, DWORD PTR _b$[ebp]
    mov     DWORD PTR _a$[ebp], eax
    mov     ecx, DWORD PTR _a$[ebp]
    push    ecx
    push    OFFSET $SG2938 ; '%d', 0aH
    call    _printf
    add     esp, 8
    pop     ebp
    ret     0
_f ENDP
```

references en C++ ([51.3 on page 723](#)), , .

64.8

...

```
#include <stdio.h>

// located in some other file
void modify_a (int *a);

void f (int a)
{
    modify_a (&a);
    printf ("%d\n", a);
}
```

Listing 64.10: Con optimización MSVC 2010

```
$SG2796 DB      '%d', 0aH, 00H

_a$ = 8
_f      PROC
    lea      eax, DWORD PTR _a$[esp-4] ; just get the ↵
    ↴ address of value in local stack
    push     eax                      ; and pass it to ↵
    ↴ modify_a()
    call     _modify_a
    mov      ecx, DWORD PTR _a$[esp]   ; reload it from the ↵
    ↴ local stack
    push     ecx                      ; and pass it to ↵
    ↴ printf()
    push     OFFSET $SG2796 ; '%d'
    call     _printf
    add      esp, 12
    ret      0
_f      ENDP
```

Modifica el valor al que apunta el puntero y luego printf() imprime el valor modificado.

Listing 64.11: Con optimización MSVC 2012 x64

```
$SG2994 DB      '%d', 0aH, 00H

a$ = 48
_f      PROC
    mov      DWORD PTR [rsp+8], ecx ; save input value in ↵
    ↴ Shadow Space
    sub      rsp, 40
    lea      rcx, QWORD PTR a$[rsp] ; get address of value ↵
    ↴ and pass it to modify_a()
    call     modify_a
    mov      edx, DWORD PTR a$[rsp] ; reload value from ↵
    ↴ Shadow Space and pass it to printf()
    lea      rcx, OFFSET FLAT:$SG2994 ; '%d'
    call     printf
    add      rsp, 40
    ret      0
_f      ENDP
```

Listing 64.12: Con optimización GCC 4.9.1 x64

```
.LC0:
.string "%d\n"
```

```
f:
    sub      rsp, 24
    mov      DWORD PTR [rsp+12], edi ; store input value to ↵
    ↳ the local stack
    lea      rdi, [rsp+12]           ; take an address of ↵
    ↳ the value and pass it to modify_a()
    call     modify_a
    mov      edx, DWORD PTR [rsp+12] ; reload value from ↵
    ↳ the local stack and pass it to printf()
    mov      esi, OFFSET FLAT:.LC0   ; '%d'
    mov      edi, 1
    xor      eax, eax
    call     __printf_chk
    add      rsp, 24
    ret
```

Listing 64.13: Con optimización GCC 4.9.1 ARM64

```
f:
    stp      x29, x30, [sp, -32]!
    add      x29, sp, 0             ; setup FP
    add      x1, x29, 32           ; calculate address of ↵
    ↳ variable in Register Save Area
    str      w0, [x1,-4]!          ; store input value there
    mov      x0, x1               ; pass address of variable ↵
    ↳ to the modify_a()
    bl      modify_a
    ldr      w1, [x29,28]          ; load value from the ↵
    ↳ variable and pass it to printf()
    adrp    x0, .LC0 ; '%d'
    add      x0, x0, :lo12:.LC0
    bl      printf                ; call printf()
    ldp      x29, x30, [sp], 32
    ret
.LC0:
    .string "%d\n"
```

: [46.1.2 on page 667](#).

Capítulo 65

Thread Local Storage

. *errno*.

C++11 *thread_local* , [TLS](#)¹:

Listing 65.1: C++11

```
#include <iostream>
#include <thread>

thread_local int tmp=3;

int main()
{
    std::cout << tmp << std::endl;
}
```

MinGW GCC 4.8.1, MSVC 2012.

tmp [TLS](#).

65.1

65.1.1 Win32

1 `#include <stdint.h>`

```

2 #include <windows.h>
3 #include <winnt.h>
4
5 // from the Numerical Recipes book:
6 #define RNG_a 1664525
7 #define RNG_c 1013904223
8
9 __declspec( thread ) uint32_t rand_state;
10
11 void my_srand (uint32_t init)
12 {
13     rand_state = init;
14 }
15
16 int my_rand ()
17 {
18     rand_state = rand_state * RNG_a;
19     rand_state = rand_state + RNG_c;
20     return rand_state & 0x7fff;
21 }
22
23 int main()
24 {
25     my_srand(0x12345678);
26     printf ("%d\n", my_rand());
27 }
```

.tls.

Listing 65.2: Con optimización MSVC 2013 x86

```

_TLS    SEGMENT
_rand_state DD 01H DUP (?)
_TLS    ENDS

_DATA   SEGMENT
$SG84851 DB      '%d', 0aH, 00H
_DATA   ENDS
_TEXT   SEGMENT

_init$ = 8                                ; size ↴
    ↓ = 4

_my_srand PROC
; FS:0=address of TIB
    mov     eax, DWORD PTR fs:_tls_array ; displayed in ↴
    ↓ IDA as FS:2Ch
; EAX=address of TLS of process
    mov     ecx, DWORD PTR __tls_index
```

```

        mov      ecx, DWORD PTR [eax+ecx*4]
; ECX=current TLS segment
        mov      eax, DWORD PTR _init$[esp-4]
        mov      DWORD PTR _rand_state[ecx], eax
        ret      0
_my_srand ENDP

_my_rand PROC
; FS:0=address of TIB
        mov      eax, DWORD PTR fs:_tls_array ; displayed in ↴
        ↴ IDA as FS:2Ch
; EAX=address of TLS of process
        mov      ecx, DWORD PTR __tls_index
        mov      ecx, DWORD PTR [eax+ecx*4]
; ECX=current TLS segment
        imul    eax, DWORD PTR _rand_state[ecx], 1664525
        add     eax, 1013904223 ; 3 ↴
        ↴ c6ef35fh
        mov      DWORD PTR _rand_state[ecx], eax
        and     eax, 32767 ; 00007 ↴
        ↴ ffff
        ret      0
_my_rand ENDP

_TEXT    ENDS

```

Listing 65.3: Con optimización MSVC 2013 x64

```

_TLS    SEGMENT
rand_state DD 01H DUP (?)
_TLS    ENDS

_DATA   SEGMENT
$SG85451 DB      '%d', 0aH, 00H
_DATA   ENDS

_TEXT   SEGMENT

init$ = 8
my_srand PROC
        mov      edx, DWORD PTR __tls_index
        mov      rax, QWORD PTR gs:88 ; 58h
        mov      r8d, OFFSET FLAT:rand_state
        mov      rax, QWORD PTR [rax+r8d*8]
        mov      DWORD PTR [r8+rax], ecx
        ret      0
my_srand ENDP

```

```

my_rand PROC
    mov     rax, QWORD PTR gs:88 ; 58h
    mov     ecx, DWORD PTR _tls_index
    mov     edx, OFFSET FLAT:rand_state
    mov     rcx, QWORD PTR [rax+rcx*8]
    imul    eax, DWORD PTR [rcx+rdx], 1664525      ; ↴
    ↳ 0019660dH
        add     eax, 1013904223                  ; ↴ 3 ↴
    ↳ c6ef35fH
        mov     DWORD PTR [rcx+rdx], eax
        and     eax, 32767                      ; 00007 ↴
    ↳ ffffH
        ret     0
my_rand ENDP

_TEXT    ENDS

```

```

1 #include <stdint.h>
2 #include <windows.h>
3 #include <winnt.h>
4
5 // from the Numerical Recipes book:
6 #define RNG_a 1664525
7 #define RNG_c 1013904223
8
9 __declspec( thread ) uint32_t rand_state=1234;
10
11 void my_srand (uint32_t init)
12 {
13     rand_state=init;
14 }
15
16 int my_rand ()
17 {
18     rand_state=rand_state*RNG_a;
19     rand_state=rand_state+RNG_c;
20     return rand_state & 0x7fff;
21 }
22
23 int main()
24 {
25     printf ("%d\n", my_rand());
26 }

```

```
.tls:00404000 ; Segment type: Pure data
.tls:00404000 ; Segment permissions: Read/Write
.tls:00404000 _tls          segment para public 'DATA' use32
.tls:00404000             assume cs:_tls
.tls:00404000             ;org 404000h
.tls:00404000 TlsStart     db    0           ; DATA  ↴
    ↳ XREF: .rdata:TlsDirectory
.tls:00404001             db    0
.tls:00404002             db    0
.tls:00404003             db    0
.tls:00404004             dd   1234
.tls:00404008 TlsEnd      db    0           ; DATA  ↴
    ↳ XREF: .rdata:TlsEnd_ptr
...
:
```

-
-
-

```
#include <stdint.h>
#include <windows.h>
#include <winnt.h>

// from the Numerical Recipes book:
#define RNG_a 1664525
#define RNG_c 1013904223

__declspec( thread ) uint32_t rand_state;

void my_srand (uint32_t init)
{
    rand_state=init;
}

void NTAPI tls_callback(PVOID a, DWORD dwReason, PVOID b)
{
    my_srand (GetTickCount());
}
```

```
#pragma data_seg(".CRT$XLB")
PIMAGE_TLS_CALLBACK p_thread_callback = tls_callback;
#pragma data_seg()

int my_rand ()
{
    rand_state=rand_state*RNG_a;
    rand_state=rand_state+RNG_c;
    return rand_state & 0x7fff;
}

int main()
{
    // rand_state is already initialized at the moment (✓
    ↳ using GetTickCount())
    printf ("%d\n", my_rand());
}
```

IDA:

Listing 65.4: Con optimización MSVC 2013

```
.text:00401020 TlsCallback_0    proc near          ; DATA ✓
    ↳ XREF: .rdata:TlsCallbacks
.text:00401020                      call    ds:GetTickCount
.text:00401026                      push    eax
.text:00401027                      call    my_srand
.text:0040102C                      pop    ecx
.text:0040102D                      retn    0Ch
.text:0040102D TlsCallback_0    endp

...
.rdata:004020C0 TlsCallbacks      dd offset TlsCallback_0 ; DATA ✓
    ↳ XREF: .rdata:TlsCallbacks_ptr

...
.rdata:00402118 TlsDirectory     dd offset TlsStart
.rdata:0040211C TlsEnd_ptr       dd offset TlsEnd
.rdata:00402120 TlsIndex_ptr     dd offset TlsIndex
.rdata:00402124 TlsCallbacks_ptr dd offset TlsCallbacks
.rdata:00402128 TlsSizeOfZeroFill dd 0
.rdata:0040212C TlsCharacteristics dd 300000h
```

65.1.2 Linux

```
__thread uint32_t rand_state=1234;
```

[2](#).

Listing 65.5: Con optimización GCC 4.8.1 x86

```
.text:08048460 my_srand          proc near
.text:08048460
.text:08048460 arg_0           = dword ptr  4
.text:08048460
.text:08048460
.text:08048464
.text:0804846A
.text:0804846A my_srand        endp

.text:08048470 my_rand          proc near
.text:08048470
    ↓ 19660Dh
.text:0804847B
.text:08048480
.text:08048486
.text:0804848B
.text:0804848B my_rand        endp
```

: [Dre13].

²<http://go.yurichev.com/17062>

Capítulo 66

- : («kernel space») («user space»).
- .
- .
- .
- . kernel panic o [BSOD¹](#).
- : ring0 («kernel space») y ring3 («user space»).
(syscall-).
- .
- .

66.1 Linux

int 0x80. EAX.

Listing 66.1:

```
section .text
global _start

_start:
    mov    edx,len ; buffer len
    mov    ecx,msg ; buffer
    mov    ebx,1    ; file descriptor. 1 is for stdout
    mov    eax,4    ; syscall number. 4 is for sys_write
    int    0x80

    mov    eax,1    ; syscall number. 4 is for sys_exit
```

¹Black Screen of Death

```
int      0x80
section .data
msg     db  'Hello, world!',0xa
len     equ $ - msg
```

```
:
```

```
nasm -f elf32 1.s
ld 1.o
```

Linux: <http://go.yurichev.com/17319>.

strace([71 on page 945](#)).

66.2 Windows

int 0x2e SYSENTER.

Windows: <http://go.yurichev.com/17320>.

```
:
```

«Windows Syscall Shellcode» by Piotr Bania:
<http://go.yurichev.com/17321>.

Capítulo 67

Linux

67.1 Código independiente de la posición

:

Listing 67.1: libc-2.17.so x86

```
.text:0012D5E3 __x86_get_pc_thunk_bx proc near ; CODE  ↴
    ↳ XREF: sub_17350+3
.text:0012D5E3                                ; ↴
    ↳ sub_173CC+4 ...
.text:0012D5E3          mov      ebx, [esp+0]
.text:0012D5E6          retn
.text:0012D5E6 __x86_get_pc_thunk_bx endp

...
.text:000576C0 sub_576C0     proc near ; CODE  ↴
    ↳ XREF: tmpfile+73
...
.text:000576C0          push    ebp
.text:000576C1          mov     ecx, large gs:0
.text:000576C8          push    edi
.text:000576C9          push    esi
.text:000576CA          push    ebx
.text:000576CB          call    __x86_get_pc_thunk_bx
.text:000576D0          add     ebx, 157930h
.text:000576D6          sub     esp, 9Ch
```

```
...
.text:000579F0          lea      eax, (a__gen_tempname - ↴
    ↴ 1AF000h)[ebx] ; "__gen_tempname"
.text:000579F6          mov      [esp+0ACh+var_A0], eax
.text:000579FA          lea      eax, (a__SysdepsPosix - ↴
    ↴ 1AF000h)[ebx] ; "../sysdeps posix/tempname.c"
.text:00057A00          mov      [esp+0ACh+var_A8], eax
.text:00057A04          lea      eax, (aInvalidKindIn_ - ↴
    ↴ 1AF000h)[ebx] ; "! \"invalid KIND in __gen_tempname\""
.text:00057A0A          mov      [esp+0ACh+var_A4], 14Ah
.text:00057A12          mov      [esp+0ACh+var_AC], eax
.text:00057A15          call    __assert_fail
```

.

:

```
#include <stdio.h>

int global_variable=123;

int f1(int var)
{
    int rt=global_variable+var;
    printf ("returning %d\n", rt);
    return rt;
}
```

[IDA](#):

```
gcc -fPIC -shared -O3 -o 1.so 1.c
```

Listing 67.2: GCC 4.7.3

```
.text:00000440          public __x86_get_pc_thunk_bx
.text:00000440 __x86_get_pc_thunk_bx proc near           ; CODE ↴
    ↴ XREF: _init_proc+4
.text:00000440           ; deregister_tm_clones+4 ...
.text:00000440           mov      ebx, [esp+0]
.text:00000443           retn
.text:00000443 __x86_get_pc_thunk_bx endp

.text:00000570          public f1
.text:00000570 f1 proc near
```

```

.text:00000570          = dword ptr -1Ch
.text:00000570 var_1C    = dword ptr -18h
.text:00000570 var_18    = dword ptr -14h
.text:00000570 var_14    = dword ptr -8
.text:00000570 var_8     = dword ptr -4
.text:00000570 var_4     = dword ptr 4
.text:00000570 arg_0     = dword ptr 4
.text:00000570
.text:00000570 sub      esp, 1Ch
.text:00000573 mov      [esp+1Ch+var_8], ebx
.text:00000577 call     __x86_get_pc_thunk_bx
.text:0000057C add      ebx, 1A84h
.text:00000582 mov      [esp+1Ch+var_4], esi
.text:00000586 mov      eax, ds:(
    ↳ global_variable_ptr - 2000h)[ebx]
.text:0000058C mov      esi, [eax]
.text:0000058E lea      eax, (aReturningD - 2000h)
    ↳ h)[ebx] ; "returning %d\n"
.text:00000594 add      esi, [esp+1Ch+arg_0]
.text:00000598 mov      [esp+1Ch+var_18], eax
.text:0000059C mov      [esp+1Ch+var_1C], 1
.text:000005A3 mov      [esp+1Ch+var_14], esi
.text:000005A7 call    __printf_chk
.text:000005AC mov      eax, esi
.text:000005AE mov      ebx, [esp+1Ch+var_8]
.text:000005B2 mov      esi, [esp+1Ch+var_4]
.text:000005B6 add      esp, 1Ch
.text:000005B9 retn
.text:000005B9 f1       endp

```

Spanish text placeholder

`__x86_get_pc_thunk_bx()` . . 0x1A84 Global Offset Table Procedure Linkage Table (GOT PLT), Global Offset Table (GOT), . [IDA](#):

```

.text:00000577 call    __x86_get_pc_thunk_bx
.text:0000057C add    ebx, 1A84h
.text:00000582 mov    [esp+1Ch+var_4], esi
.text:00000586 mov    eax, [ebx-0Ch]
.text:0000058C mov    esi, [eax]
.text:0000058E lea    eax, [ebx-1A30h]

```

. .
. .
. .
:

```

00000000000000720 <f1>:
720: 48 8b 05 b9 08 20 00    mov    rax,QWORD PTR [rip+0x20]
     ↳ x2008b9]          # 200fe0 <_DYNAMIC+0x1d0>
727: 53                      push   rbx
728: 89 fb                  mov    ebx,edi
72a: 48 8d 35 20 00 00 00    lea    rsi,[rip+0x20]      # 
     ↳ 751 <_fini+0x9>
731: bf 01 00 00 00          mov    edi,0x1
736: 03 18                  add    ebx,DWORD PTR [rax]
738: 31 c0                  xor    eax,eax
73a: 89 da                  mov    edx,ebx
73c: e8 df fe ff ff          call   620 <__printf_chk@plt>
741: 89 d8                  mov    eax,ebx
743: 5b                      pop   rbx
744: c3                      ret

```

0x2008b9 .

. [Fog13a].

67.1.1 Windows

67.2 *LD_PRELOAD* en Linux

libc.so.6.

time(), read(), write(), Spanish text placeholder.

.. strace([71 on page 945](#)), /proc/uptime :

```
$ strace uptime
...
open("/proc/uptime", O_RDONLY)      = 3
lseek(3, 0, SEEK_SET)              = 0
read(3, "416166.86 414629.38\n", 2047) = 20
...
```

:

```
$ cat /proc/uptime
416690.91 415152.03
```

[1](#):

¹wikipedia

The first number is the total number of seconds the system has been up. The second number is how much of that time the machine has spent idle, in seconds.

open(), read(), close().
read() libc.so.6. close(),
[2](#).

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <dlfcn.h>
#include <string.h>

void *libc_handle = NULL;
int (*open_ptr)(const char *, int) = NULL;
int (*close_ptr)(int) = NULL;
ssize_t (*read_ptr)(int, void*, size_t) = NULL;

bool initied = false;

_Noreturn void die (const char * fmt, ...)
{
    va_list va;
    va_start (va, fmt);

    vprintf (fmt, va);
    exit(0);
};

static void find_original_functions ()
{
    if (initied)
        return;

    libc_handle = dlopen ("libc.so.6", RTLD_LAZY);
    if (libc_handle==NULL)
        die ("can't open libc.so.6\n");

    open_ptr = dlsym (libc_handle, "open");
    if (open_ptr==NULL)
```

²en [3](#) Yong Huang

```
        die ("can't find open()\n");

    close_ptr = dlsym (libc_handle, "close");
    if (close_ptr==NULL)
        die ("can't find close()\n");

    read_ptr = dlsym (libc_handle, "read");
    if (read_ptr==NULL)
        die ("can't find read()\n");

    initied = true;
}

static int opened_fd=0;

int open(const char *pathname, int flags)
{
    find_original_functions();

    int fd=(*open_ptr)(pathname, flags);
    if (strcmp(pathname, "/proc/uptime")==0)
        opened_fd=fd; // that's our file! record its ↴
    ↵ file descriptor
    else
        opened_fd=0;
    return fd;
};

int close(int fd)
{
    find_original_functions();

    if (fd==opened_fd)
        opened_fd=0; // the file is not opened anymore
    return (*close_ptr)(fd);
};

ssize_t read(int fd, void *buf, size_t count)
{
    find_original_functions();

    if (opened_fd!=0 && fd==opened_fd)
    {
        // that's our file!
        return sprintf (buf, count, "%d %d", 0↖
    ↵ x7fffffff, 0x7fffffff)+1;
    };
}
```

```
// not our file, go to real read() function  
return (*read_ptr)(fd, buf, count);  
};
```

(GitHub)

:

```
gcc -fPIC -shared -Wall -o fool_uptime.so fool_uptime.c -ldl
```

:

```
LD_PRELOAD=`pwd`/fool_uptime.so uptime
```

:

```
01:23:02 up 24855 days, 3:14, 3 users, load average: 0.00, ↴  
↳ 0.01, 0.05
```

LD_PRELOAD

:

- strcmp() (Yong Huang) <http://go.yurichev.com/17043>
- Kevin PuloFun with LD_PRELOAD..yurichev.com
- (zlibc). <http://go.yurichev.com/17146>

Capítulo 68

Windows NT

68.1 CRT (win32)

```
?.. .
```

```
...:
```

```
int CALLBACK WinMain(
    _In_ HINSTANCE hInstance,
    _In_ HINSTANCE hPrevInstance,
    _In_ LPSTR lpCmdLine,
    _In_ int nCmdShow
);
```

```
.
```

```
..
```

```
.
```

```
.
```

```
1 __tmainCRTStartup proc near
2
3 var_24 = dword ptr -24h
4 var_20 = dword ptr -20h
5 var_1C = dword ptr -1Ch
6 ms_exc = CPPEH_RECORD ptr -18h
7
8     push    14h
```

```
9      push    offset stru_4092D0
10     call    __SEH_prolog4
11     mov     eax, 5A4Dh
12     cmp     ds:400000h, ax
13     jnz    short loc_401096
14     mov     eax, ds:40003Ch
15     cmp     dword ptr [eax+400000h], 4550h
16     jnz    short loc_401096
17     mov     ecx, 10Bh
18     cmp     [eax+400018h], cx
19     jnz    short loc_401096
20     cmp     dword ptr [eax+400074h], 0Eh
21     jbe    short loc_401096
22     xor     ecx, ecx
23     cmp     [eax+4000E8h], ecx
24     setnz  cl
25     mov     [ebp+var_1C], ecx
26     jmp    short loc_40109A
27
28
29 loc_401096: ; CODE XREF: __tmainCRTStartup+18
30             ; __tmainCRTStartup+29 ...
31     and    [ebp+var_1C], 0
32
33 loc_40109A: ; CODE XREF: __tmainCRTStartup+50
34     push   1
35     call    __heap_init
36     pop    ecx
37     test   eax, eax
38     jnz    short loc_4010AE
39     push   1Ch
40     call    _fast_error_exit
41     pop    ecx
42
43 loc_4010AE: ; CODE XREF: __tmainCRTStartup+60
44     call    __mtinit
45     test   eax, eax
46     jnz    short loc_4010BF
47     push   10h
48     call    _fast_error_exit
49     pop    ecx
50
51 loc_4010BF: ; CODE XREF: __tmainCRTStartup+71
52     call    sub_401F2B
53     and    [ebp+ms_exc.disabled], 0
54     call    __ioinit
55     test   eax, eax
```

```
56      jge    short loc_4010D9
57      push   1Bh
58      call   __amsg_exit
59      pop    ecx
60
61 loc_4010D9: ; CODE XREF: __tmainCRTStartup+8B
62      call   ds:GetCommandLineA
63      mov    dword_40B7F8, eax
64      call   __crtGetEnvironmentStringsA
65      mov    dword_40AC60, eax
66      call   __setargv
67      test  eax, eax
68      jge   short loc_4010FF
69      push  8
70      call   __amsg_exit
71      pop   ecx
72
73 loc_4010FF: ; CODE XREF: __tmainCRTStartup+B1
74      call   __setenvp
75      test  eax, eax
76      jge   short loc_401110
77      push  9
78      call   __amsg_exit
79      pop   ecx
80
81 loc_401110: ; CODE XREF: __tmainCRTStartup+C2
82      push  1
83      call   __cinit
84      pop   ecx
85      test  eax, eax
86      jz    short loc_401123
87      push  eax
88      call   __amsg_exit
89      pop   ecx
90
91 loc_401123: ; CODE XREF: __tmainCRTStartup+D6
92      mov   eax, envp
93      mov   dword_40AC80, eax
94      push  eax          ; envp
95      push  argv          ; argv
96      push  argc          ; argc
97      call   _main
98      add   esp, 0Ch
99      mov   [ebp+var_20], eax
100     cmp   [ebp+var_1C], 0
101     jnz   short $LN28
102     push  eax          ; uExitCode
```

```

103          call    $LN32
104
105 $LN28:      ; CODE XREF: __tmainCRTStartup+105
106         call    __cexit
107         jmp     short loc_401186
108
109
110 $LN27:      ; DATA XREF: .rdata:stru_4092D0
111         mov    eax, [ebp+ms_exc.exc_ptr] ; Exception filter 0 ↴
112         ↳ for function 401044
113         mov    ecx, [eax]
114         mov    ecx, [ecx]
115         mov    [ebp+var_24], ecx
116         push   eax
117         push   ecx
118         call   __XcptFilter
119         pop    ecx
120         pop    ecx
121
122 $LN24:
123         retn
124
125
126 $LN14:      ; DATA XREF: .rdata:stru_4092D0
127         mov    esp, [ebp+ms_exc.old_esp] ; Exception handler 0 ↴
128         ↳ for function 401044
129         mov    eax, [ebp+var_24]
130         mov    [ebp+var_20], eax
131         cmp    [ebp+var_1C], 0
132         jnz   short $LN29
133         push   eax           ; int
134         call   __exit
135
136 $LN29:      ; CODE XREF: __tmainCRTStartup+135
137         call   __c_exit
138
139 loc_401186: ; CODE XREF: __tmainCRTStartup+112
140         mov    [ebp+ms_exc.disabled], 0FFFFFFFEh
141         mov    eax, [ebp+var_20]
142         call   __SEH_epilog4
143         retn

```

GetCommandLineA() (línea 62), setargv() (línea 66) y setenvp() (línea 74), argc, argv, envp. (línea 97).

`heap_init()` (línea 35), `ioinit()` (línea 54).

. `malloc()` CRT, :

```
runtime error R6030
- CRT not initialized
```

: 51.4.1 on page 730.

`main()` `cexit()`, `$LN32`, `doexit()`.

```
#include <windows.h>

int main()
{
    MessageBox (NULL, "hello, world", "caption", MB_OK);
}
```

MSVC 2008.

```
cl no_crt.c user32.lib /link /entry:main
```

```
cl no_crt.c user32.lib /link /entry:main /align:16
```

:

```
LINK : warning LNK4108: /ALIGN specified without /DRIVER; image
      ↴ may not run
```

. Windows 7 x86, x64 ()�

68.2 Win32 PE

PE es un formato de archivo ejecutable usado en Windows.

La diferencia entre .exe, .dll y .sys es que .exe y .sys normalmente no tienen exportaciones, solo importaciones.

Una **DLL**¹, igual que cualquier otro archivo PE, tiene un punto de entrada (**OEP**²) (allí está la función `DllMain()`), pero normalmente esa función no hace nada.

¹Dynamic-link library

²Original Entry Point

.sys normalmente es un controlador de dispositivo.

Para los controladores, Windows exige que la suma de comprobación esté presente en el archivo PE y sea correcta ³.

Windows Vista, Desde Windows Vista, los archivos de los controladores también deben ir firmados digitalmente; de lo contrario no se cargarán.

«This program cannot be run in DOS mode.» Al comienzo de cualquier archivo PE hay un pequeño programa DOS que muestra un mensaje como «This program cannot be run in DOS mode.» si se ejecuta en DOS o Windows 3.1 ([OS](#) que no conocen el formato PE), se mostrará ese mensaje.

68.2.1

- Módulos un archivo independiente, .exe o .dll.
- Procesos un programa cargado en memoria y ejecutándose. Normalmente consta de un archivo .exe y un montón de archivos .dll.
- Memoria del proceso la memoria con la que trabaja el proceso. Cada proceso tiene la suya. Suele haber módulos cargados, memoria de la pila, [heap\(s\)](#), Spanish text placeholder.
- [VA](#)⁴ es una dirección que se usa en el propio programa en tiempo de ejecución.
- es la dirección en la memoria del proceso donde debe cargarse el módulo. El cargador de la [OS](#) puede cambiarla si ese destino ya está ocupado por otro módulo cargado antes.
- [RVA](#)⁵ [VA](#)-es la dirección [VA](#) menos la dirección base. [RVA](#)-Muchas direcciones en las tablas de un archivo PE usan direcciones [RVA](#).
- [IAT](#)⁶ un arreglo de direcciones de símbolos importados ⁷. [IMAGE_DIRECTORY_ENTRY_IAT](#) veces, la entrada de directorio de datos [IMAGE_DIRECTORY_ENTRY_IAT](#) apunta a la [IAT](#). [IDA](#) (6.1) .idata para [IAT](#), [IAT](#) !Es importante notar que [IDA](#) (al menos hasta la 6.1) puede crear una seudosección llamada .idata para la [IAT](#), incluso si la [IAT](#) forma parte de otra sección!
- [INT](#)⁸ un arreglo con los nombres de los símbolos a importar⁹.

³Por ejemplo, Hiew([73 on page 947](#)) puede calcularla

⁴Virtual Address

⁵Relative Virtual Address

⁶Import Address Table

⁷[[Pie02](#)]

⁸Import Name Table

⁹[[Pie02](#)]

68.2.2

El problema es que varios autores de módulos pueden preparar DLL para que otros las usen y no es posible acordar qué direcciones se asignan a quién.

Por eso, si dos DLL necesarias para un proceso tienen la misma dirección base, una se cargará en esa base y la otra en otro espacio libre de la memoria del proceso, y todas las direcciones virtuales de la segunda DLL se ajustarán.

[MSVC](#) 0x400000¹⁰, 0x401000. [RVA](#) 0x1000. 0x10000000¹¹ Con frecuencia, el enlazador de [MSVC](#) genera archivos .exe con dirección base 0x400000 y la sección de código arrancando en 0x401000; esto significa que el [RVA](#) del inicio de la sección de código es 0x1000. Las [DLL](#) suelen generarse con base 0x10000000, y se puede cambiar con la opción /BASE del enlazador.

Además hay otra razón para cargar los módulos en direcciones distintas, concretamente en direcciones aleatorias.

[ASLR](#)¹²¹³ Esto es [ASLR](#).

Un shellcode que intenta ejecutarse en un sistema comprometido debe llamar funciones del sistema y, por lo tanto, conocer sus direcciones.

[OS](#) (Windows Vista), [DLL](#) (kernel32.dll, user32.dll) , , las DLL del sistema (como kernel32.dll o user32.dll) se cargaban siempre en direcciones conocidas y, si recordamos que sus versiones cambiaban rara vez, las direcciones de las funciones eran prácticamente fijas y el shellcode podía llamarlas directamente.

[ASLR](#) carga tanto tu programa como todos los módulos que necesita en direcciones base aleatorias, diferentes en cada ejecución.

El soporte de [ASLR](#) en un archivo PE se marca activando el indicador IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE [[RA09](#)].

68.2.3 Subsystem

También hay un campo *subsystem*, que suele ser:

- native¹⁴ (.sys-controlador),
- console (aplicación de consola) o
- GUI¹⁵ (no es de consola).

¹⁰. [MSDN](#)

¹¹

¹² Address Space Layout Randomization

¹³

¹⁴ Significa que el módulo usa Native API en lugar de Win32

¹⁵ Graphical user interface

68.2.4

El archivo PE también especifica el número mínimo de versión de Windows necesario para cargar el módulo. aquí¹⁶.

, MSVC 2005 Windows NT4 (4.00), MSVC 2008 (5.00, los archivos generados tienen versión 5.00 y requieren al menos Windows 2000 para ejecutarse).

Por defecto, MSVC 2012 genera archivos .exe de versión 6.00, que necesitan al menos Windows Vista; sin embargo, cambiando las opciones del compilador¹⁷ es posible forzar la generación para Windows XP.

68.2.5

- *IMAGE_SCN_CNT_CODE* o *IMAGE_SCN_MEM_EXECUTE*.
- *IMAGE_SCN_CNT_INITIALIZED_DATA*, *IMAGE_SCN_MEM_READ* y *IMAGE_SCN_MEM_WRITE*
- *IMAGE_SCN_CNT_UNINITIALIZED_DATA*, *IMAGE_SCN_MEM_READ* y *IMAGE_SCN_MEM_WRITE*
- , En una sección de datos constantes (protegida contra escritura) pueden estar los indicadores

IMAGE_SCN_CNT_INITIALIZED_DATA y *IMAGE_SCN_MEM_READ* *IMAGE_SCN_MEM_WRITE*
Si se intenta escribir algo en esa sección, el proceso fallará.

En un archivo PE se puede asignar un nombre a cada sección, aunque no es algo crucial. .text la sección de código se llama .text, .data, – .rdata (*readable data*) (datos legibles). Otros nombres comunes de secciones son:

- .idata sección de importaciones. IDA puede crear una sección con el mismo nombre: [68.2.1 on page 900](#).
- .edata sección de exportaciones (poco común)
- .pdata sección que contiene toda la información sobre excepciones en Windows NT para MIPS, IA64 y x64: [68.3.3 on page 933](#)
- .reloc sección de relocs
- .bss datos no inicializados ([BSS¹⁸](#))
- .tls thread local storage ([TLS](#)) almacenamiento local de hilo ([TLS](#))
- .rsrc recursos
- .CRT puede aparecer en binarios compilados con versiones muy antiguas de MSVC

¹⁶[wikipedia](#)

¹⁷[MSDN](#)

¹⁸Block Started by Symbol

Los empaquetadores o cifradores de archivos PE suelen borrar los nombres de las secciones o cambiarlos por los suyos propios.

En [MSVC](#) se pueden declarar datos en una sección con nombre arbitrario ¹⁹.

Algunos compiladores y enlazadores pueden añadir también una sección con símbolos de depuración y, en general, información de depuración (por ejemplo, MinGW). [MSVC](#) ()allí se suele guardar la información de depuración en archivos [PDB](#) separados.

Así se describe una sección PE en el archivo:

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
        DWORD PhysicalAddress;
        DWORD VirtualSize;
    } Misc;
    DWORD VirtualAddress;
    DWORD SizeOfRawData;
    DWORD PointerToRawData;
    DWORD PointerToRelocations;
    DWORD PointerToLinenumbers;
    WORD NumberOfRelocations;
    WORD NumberOfLinenumbers;
    DWORD Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

²⁰

Algo más de terminología: *PointerToRawData* «Offset» en Hiew y *VirtualAddress* «RVA» allí.

68.2.6

(Hiew).

²¹.

. código independiente de la posición ([67.1 on page 888](#)). .

0x410000 :

A1 00 00 41 00	mov	eax, [000410000]
----------------	-----	------------------

¹⁹[MSDN](#)

²⁰[MSDN](#)

²¹MS-DOS

0x400000, [RVA](#) 0x10000.

0x500000, 0x510000.

MOV, 0xA1.

0xA1, .

, , , ([RVA](#)), .

.

.

,

IMAGE_REL_BASED_HIGHLOW.

: Spanish text placeholder.[7.12](#).

OllyDbg: Spanish text placeholder.[13.11](#).

68.2.7

,

.

.

.

.

.

, [MFC²²](#), .

. *mfc80_123.*

²²Microsoft Foundation Classes

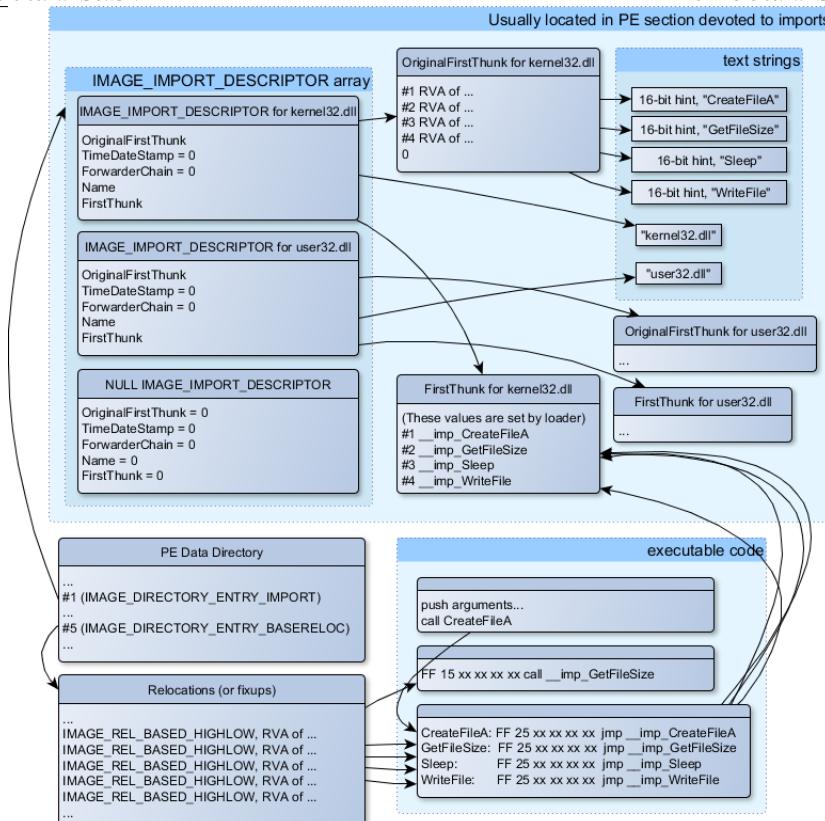


Figura 68.1:

IMAGE_IMPORT_DESCRIPTOR..

RVA (Name).

OriginalFirstThink RVA . RVA, . «hint».

FirstThunk, RVA .

:_imp_CreateFileA, Spanish text placeholder.

- call __imp_CreateFileA, , , .

...

-

. PE_add_import²³.

```
MOV EAX, [yourdll.dll!function]
JMP EAX
```

,
IMAGE_SCN_MEM_WRITE 5 (access denied).

IMAGE_IMPORT_DESCRIPTOR . . «old-style binding»²⁴ . . , Matt Pietrek en [[Pie02](#)],

. . *LoadLibrary()* y *GetProcAddress()*.

68.2.8

([68.2.11 on the next page](#)).

68.2.9 .NET

```
jmp mscoree.dll!_CorExeMain
```

²⁵

68.2.10 TLS

[TLS\(65 on page 879\)](#) () . **TLS** .

[TLS](#) TLS callbacks. ([OEP](#)).

²³[yurichev.com](#)

²⁴[MSDN](#). «new-style binding».

²⁵[MSDN](#)

68.2.11

- objdump (cygwin) .
- Hiew([73 on page 947](#)) .
- pefilePython- ²⁶.
- ResHack AKA Resource Hacker ²⁷.
- PE_add_import²⁸ .
- PE_patcher²⁹ .
- PE_search_str_refs³⁰ .

68.2.12 Further reading

- Daniel PistelliThe .NET File Format ³¹

68.3 Windows SEH

68.3.1

. . .
. . .
. .

²⁶<http://go.yurichev.com/17052>

²⁷<http://go.yurichev.com/17052>

²⁸<http://go.yurichev.com/17049>

²⁹yurichev.com

³⁰yurichev.com

³¹<http://go.yurichev.com/17056>

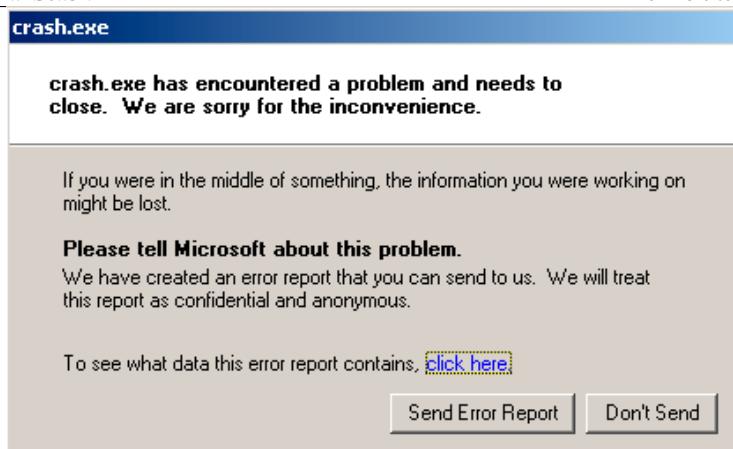


Figura 68.2: Windows XP

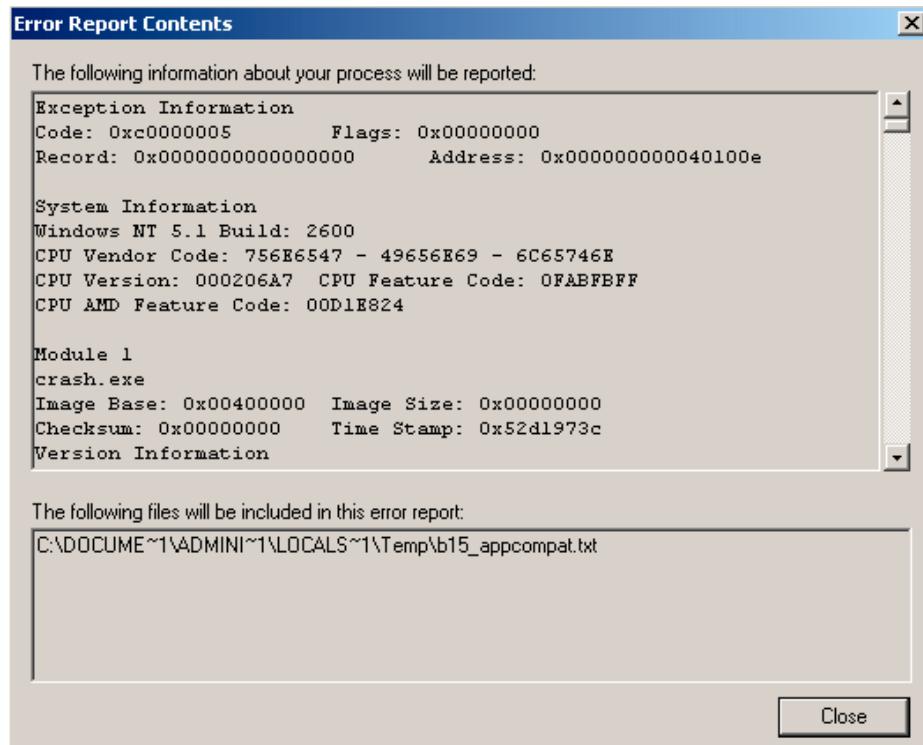


Figura 68.3: Windows XP

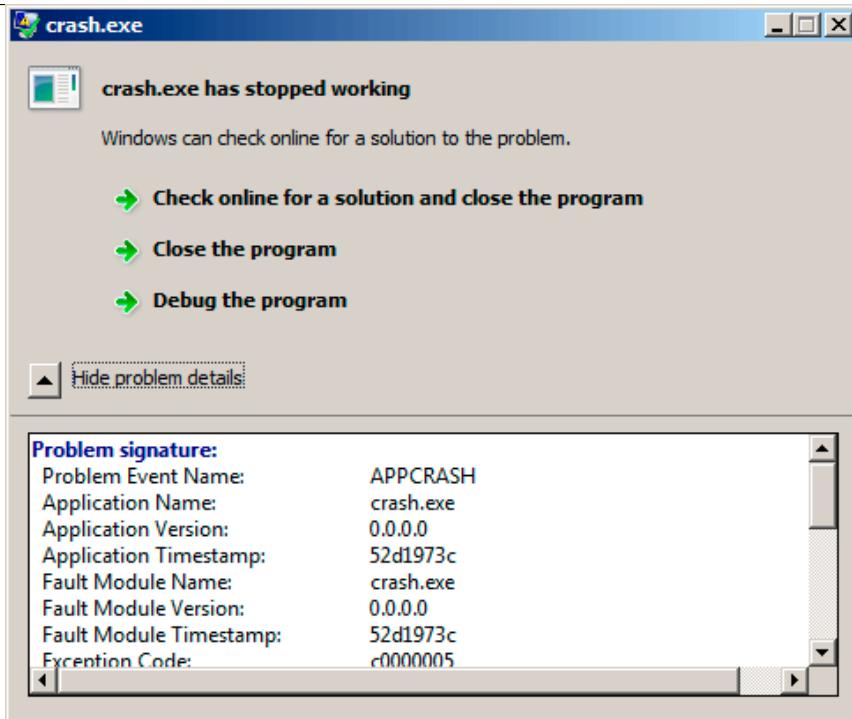


Figura 68.4: Windows 7

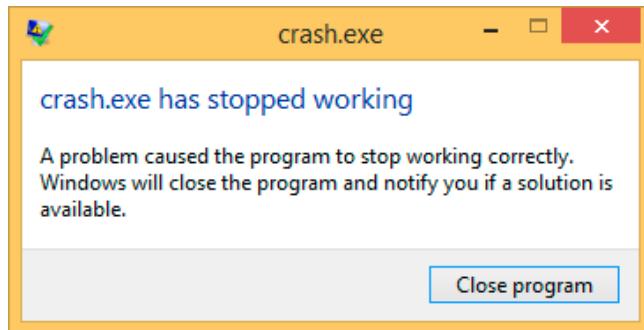


Figura 68.5: Windows 8.1

.SetUnhandledExceptionFilter() Oracle RDBMS .

33:

```
#include <windows.h>
#include <stdio.h>

DWORD new_value=1234;

EXCEPTION_DISPOSITION __cdecl except_handler(
    struct _EXCEPTION_RECORD *ExceptionRecord,
    void * EstablisherFrame,
    struct _CONTEXT *ContextRecord,
    void * DispatcherContext )
{
    unsigned i;

    printf ("%s\n", __FUNCTION__);
    printf ("ExceptionRecord->ExceptionCode=0x%p\n",
        ↴ ExceptionRecord->ExceptionCode);
    printf ("ExceptionRecord->ExceptionFlags=0x%p\n",
        ↴ ExceptionRecord->ExceptionFlags);
    printf ("ExceptionRecord->ExceptionAddress=0x%p\n",
        ↴ ExceptionRecord->ExceptionAddress);

    if (ExceptionRecord->ExceptionCode==0xE1223344)
    {
        printf ("That's for us\n");
        // yes, we "handled" the exception
        return ExceptionContinueExecution;
    }
    else if (ExceptionRecord->ExceptionCode==
        ↴ EXCEPTION_ACCESS_VIOLATION)
    {
        printf ("ContextRecord->Eax=0x%08X\n",
            ↴ ContextRecord->Eax);
        // will it be possible to 'fix' it?
        printf ("Trying to fix wrong pointer address\n
        ↴ ");
        ContextRecord->Eax=(DWORD)&new_value;
        // yes, we "handled" the exception
        return ExceptionContinueExecution;
    }
    else
```

³³[Pie]

SAFESEH: cl seh1.cpp /link /safeseh:no

SAFESEH :

MSDN

```

    {
        printf ("We do not handle this\n");
        // someone else's problem
        return ExceptionContinueSearch;
    };
}

int main()
{
    DWORD handler = (DWORD)except_handler; // take a ↴
    ↴ pointer to our handler

    // install exception handler
    __asm
    {
        ↴ EXCEPTION_REGISTRATION record:           // make ↴
        push    handler                         // address of handler ↴
    ↴ function
        push    FS:[0]                          // address of previous ↴
    ↴ handler
        mov     FS:[0],ESP                   // add new ↴
    ↴ EXCEPTION_REGISTRATION
    }

    RaiseException (0xE1223344, 0, 0, NULL);

    // now do something very bad
    int* ptr=NULL;
    int val=0;
    val=*ptr;
    printf ("val=%d\n", val);

    // deinstall exception handler
    __asm
    {
        ↴ EXCEPTION_REGISTRATION record           // remove our ↴
        mov     eax,[ESP]                      // get pointer to ↴
    ↴ previous record
        mov     FS:[0], EAX                  // install previous ↴
    ↴ record
        add     esp, 8                     // clean our ↴
    ↴ EXCEPTION_REGISTRATION off stack
    }

    return 0;
}

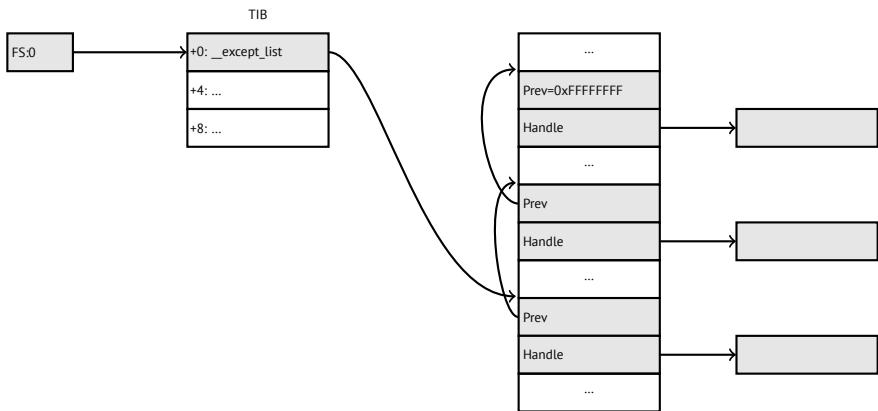
```

. . . _EXCEPTION_REGISTRATION, .

Listing 68.1: MSVC/NC/crt/src/exsup.inc

```
\_EXCEPTION\_REGISTRATION struc
    prev    dd      ?
    handler dd      ?
\_EXCEPTION\_REGISTRATION ends
```

«handler» «prev» . 0xFFFFFFFF (-1) «prev».

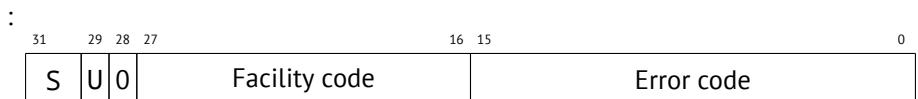


`RaiseException()`³⁴ . . . 0xE1223344, `ExceptionContinueExecution`,
. `ExceptionContinueSearch`, . . .

? :

³⁴[MSDN](#)

WinBase.h	ntstatus.h
EXCEPTION_ACCESS_VIOLATION	STATUS_ACCESS_VIOLATION
EXCEPTION_DATATYPE_MISALIGNMENT	STATUS_DATATYPE_MISALIGNMENT
EXCEPTION_BREAKPOINT	STATUS_BREAKPOINT
EXCEPTION_SINGLE_STEP	STATUS_SINGLE_STEP
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	STATUS_ARRAY_BOUNDS_EXCEEDED
EXCEPTION_FLT_DENORMAL_OPERAND	STATUS_FLOAT_DENORMAL_OPERAND
EXCEPTION_FLT_DIVIDE_BY_ZERO	STATUS_FLOAT_DIVIDE_BY_ZERO
EXCEPTION_FLT_INEXACT_RESULT	STATUS_FLOAT_INEXACT_RESULT
EXCEPTION_FLT_INVALID_OPERATION	STATUS_FLOAT_INVALID_OPERATION
EXCEPTION_FLT_OVERFLOW	STATUS_FLOAT_OVERFLOW
EXCEPTION_FLT_STACK_CHECK	STATUS_FLOAT_STACK_CHECK
EXCEPTION_FLT_UNDERFLOW	STATUS_FLOAT_UNDERFLOW
EXCEPTION_INT_DIVIDE_BY_ZERO	STATUS_INTEGER_DIVIDE_BY_ZERO
EXCEPTION_INT_OVERFLOW	STATUS_INTEGER_OVERFLOW
EXCEPTION_PRIV_INSTRUCTION	STATUS_PRIVILEGED_INSTRUCTION
EXCEPTION_IN_PAGE_ERROR	STATUS_IN_PAGE_ERROR
EXCEPTION_ILLEGAL_INSTRUCTION	STATUS_ILLEGAL_INSTRUCTION
EXCEPTION_NONCONTINUABLE_EXCEPTION	STATUS_NONCONTINUABLE_EXCEPTION
EXCEPTION_STACK_OVERFLOW	STATUS_STACK_OVERFLOW
EXCEPTION_INVALID_DISPOSITION	STATUS_INVALID_DISPOSITION
EXCEPTION_GUARD_PAGE	STATUS_GUARD_PAGE_VIOLATION
EXCEPTION_INVALID_HANDLE	STATUS_INVALID_HANDLE
EXCEPTION_POSSIBLE_DEADLOCK	STATUS_POSSIBLE_DEADLOCK
CONTROL_C_EXIT	STATUS_CONTROL_C_EXIT



S : 11; 10; 01; 00. U.

0xE1223344 0xE (1110b) . .

. . .

:

Listing 68.2: MSVC 2010

```

...
xor    eax, eax
mov    eax, DWORD PTR [eax] ; exception will occur ↴
↳ here
push   eax
push   OFFSET msg
call   _printf

```

```
add    esp, 8
...
...
```

? .. printf() 1234, EAX 0, new_value..

; , , , .

? alloca(): ([5.2.4 on page 36](#)).

68.3.2

[35](#) ..

```
__try
{
    ...
}
__except(filter code)
{
    handler code
}
```

:

```
__try
{
    ...
}
__finally
{
    ...
}
```

..

. ([WRK](#)):

Listing 68.3: WRK-v1.2/base/ntos/ob/obwait.c

```
try {
    KeReleaseMutant( (PKMUTANT)SignalObject,
                      MUTANT_INCREMENT,
                      FALSE,
                      TRUE );
```

³⁵[MSDN](#)

```

} except((GetExceptionCode () == STATUS_ABANDONED ||
          GetExceptionCode () == STATUS_MUTANT_NOT_OWNED)?
          EXCEPTION_EXECUTE_HANDLER :
          EXCEPTION_CONTINUE_SEARCH) {
    Status = GetExceptionCode();
    goto WaitExit;
}

```

Listing 68.4: WRK-v1.2/base/ntos/cache/cachesub.c

```

try {

    RtlCopyBytes( (PVOID)((PCHAR)CacheBuffer + PageOffset),
                  UserBuffer,
                  MorePages ?
                  (PAGE_SIZE - PageOffset) :
                  (ReceivedLength - PageOffset) );

} except( CcCopyReadExceptionFilter( GetExceptionInformation(),
                                    & Status ) ) {

```

:

Listing 68.5: WRK-v1.2/base/ntos/cache/copysup.c

```

LONG
CcCopyReadExceptionFilter(
    IN PEXCEPTION_POINTERS ExceptionPointer,
    IN PNTSTATUS ErrorCode
)

```

/*++

Routine Description:

This routine serves as a exception filter and has the ↴
 ↳ special job of
 extracting the "real" I/O error when Mm raises ↴
 ↳ STATUS_IN_PAGE_ERROR
 beneath us.

Arguments:

ExceptionPointer - A pointer to the exception record that ↴
 ↳ contains
 the real Io Status.

```
ExceptionCode - A pointer to an NTSTATUS that is to receive
↳ the real
                status.
```

Return Value:

```
EXCEPTION_EXECUTE_HANDLER
--*/
{
    *ExceptionCode = ExceptionPointer->ExceptionRecord->
↳ ExceptionCode;

    if ( (*ExceptionCode == STATUS_IN_PAGE_ERROR) &&
        (ExceptionPointer->ExceptionRecord->NumberParameters
         >= 3) ) {

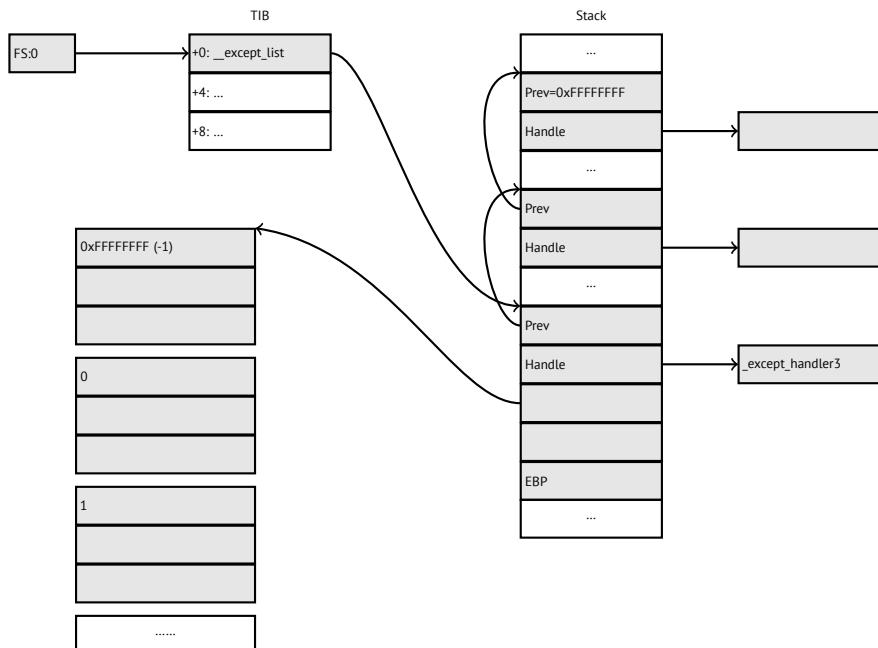
        *ExceptionCode = (NTSTATUS) ExceptionPointer->
↳ ExceptionRecord->ExceptionInformation[2];
    }

    ASSERT( !NT_SUCCESS(*ExceptionCode) );
    return EXCEPTION_EXECUTE_HANDLER;
}
```

. _except_handler3 (para SEH3) o _except_handler4 (para SEH4). msvcr*.dll.

SEH3

SEH3 _except_handler3 _EXCEPTION_REGISTRATION_SEH4 *scope table* .
scope table .



prev/handle . _except_handler3 .

_except_handler3 . ³⁶ Wine ³⁷ y ReactOS ³⁸.

filter handler .

SEH3:

```
#include <stdio.h>
#include <windows.h>
#include <excpt.h>

int main()
{
    int* p = NULL;
    __try
    {
```

³⁶<http://go.yurichev.com/17058>

³⁷[GitHub](#)

³⁸<http://go.yurichev.com/17060>

```

        printf("hello #1!\n");
        *p = 13;      // causes an access violation exception;
        printf("hello #2!\n");
    }
__except(GetExceptionCode()==EXCEPTION_ACCESS_VIOLATION ?
    EXCEPTION_EXECUTE_HANDLER : ↴
    EXCEPTION_CONTINUE_SEARCH)
{
    printf("access violation, can't recover\n");
}
}

```

Listing 68.6: MSVC 2003

```

$SG74605 DB      'hello #1!', 0aH, 00H
$SG74606 DB      'hello #2!', 0aH, 00H
$SG74608 DB      'access violation, can't recover', 0aH, 00H
_DATA    ENDS

; scope table:
CONST    SEGMENT
$T74622  DD      0xffffffffH      ; previous try level
           DD      FLAT:$L74617   ; filter
           DD      FLAT:$L74618   ; handler

CONST    ENDS
_TEXT    SEGMENT
$T74621 = -32 ; size = 4
_p$ = -28   ; size = 4
__$SEHRec$ = -24 ; size = 24
_main    PROC NEAR
    push    ebp
    mov     ebp, esp
    push    -1                     ; previous try level
    push    OFFSET FLAT:$T74622    ; scope table
    push    OFFSET FLAT:_except_handler3 ; handler
    mov     eax, DWORD PTR fs:_except_list
    push    eax                     ; prev
    mov     DWORD PTR fs:_except_list, esp
    add     esp, -16
; 3 registers to be saved:
    push    ebx
    push    esi
    push    edi
    mov     DWORD PTR __$SEHRec$[ebp], esp
    mov     DWORD PTR _p$[ebp], 0
    mov     DWORD PTR __$SEHRec$[ebp+20], 0 ; previous try ↴
    ↴ level

```

```

push  OFFSET FLAT:$SG74605 ; 'hello #1!'
call  _printf
add   esp, 4
mov   eax, DWORD PTR _p$[ebp]
mov   DWORD PTR [eax], 13
push  OFFSET FLAT:$SG74606 ; 'hello #2!'
call  _printf
add   esp, 4
mov   DWORD PTR __$SEHRec$[ebp+20], -1 ; previous try ↴
↳ level
jmp   SHORT $L74616

; filter code:
$L74617:
$L74627:
    mov   ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov   edx, DWORD PTR [ecx]
    mov   eax, DWORD PTR [edx]
    mov   DWORD PTR $T74621[ebp], eax
    mov   eax, DWORD PTR $T74621[ebp]
    sub   eax, -1073741819; c0000005H
    neg   eax
    sbb   eax, eax
    inc   eax
$L74619:
$L74626:
    ret   0

; handler code:
$L74618:
    mov   esp, DWORD PTR __$SEHRec$[ebp]
    push  OFFSET FLAT:$SG74608 ; 'access violation, can''t ↴
    ↳ recover'
    call  _printf
    add   esp, 4
    mov   DWORD PTR __$SEHRec$[ebp+20], -1 ; setting previous ↴
    ↳ try level back to -1
$L74616:
    xor   eax, eax
    mov   ecx, DWORD PTR __$SEHRec$[ebp+8]
    mov   DWORD PTR fs:_except_list, ecx
    pop   edi
    pop   esi
    pop   ebx
    mov   esp, ebp
    pop   ebp
    ret   0

```

```
_main    ENDP
_TEXT    ENDS
END
```

. CONST. previous try level. 0xFFFFFFFF (-1). try 0. try , -1...

SEH_prolog()..

tracer:

```
tracer.exe -l:2.exe --dump-seh
```

Listing 68.7: tracer.exe output

```
EXCEPTION_ACCESS_VIOLATION at 2.exe!main+0x44 (0x401054) ↴
  ↳ ExceptionInformation[0]=1
EAX=0x00000000 EBX=0x7efde000 ECX=0x0040cbc8 EDX=0x0008e3c8
ESI=0x00001db1 EDI=0x00000000 EBP=0x0018feac ESP=0x0018fe80
EIP=0x00401054
FLAGS=AF IF RF
* SEH frame at 0x18fe9c prev=0x18ff78 handler=0x401204 (2.exe! ↴
  ↳ _except_handler3)
SEH3 frame. previous trylevel=0
scopetable entry[0]. previous try level=-1, filter=0x401070 (2. ↴
  ↳ exe!main+0x60) handler=0x401088 (2.exe!main+0x78)
* SEH frame at 0x18ff78 prev=0x18ffc4 handler=0x401204 (2.exe! ↴
  ↳ _except_handler3)
SEH3 frame. previous trylevel=0
scopetable entry[0]. previous try level=-1, filter=0x401531 (2. ↴
  ↳ exe!mainCRTStartup+0x18d) handler=0x401545 (2.exe! ↴
  ↳ mainCRTStartup+0x1a1)
* SEH frame at 0x18ffc4 prev=0x18ffe4 handler=0x771f71f5 (ntdll. ↴
  ↳ .dll!__except_handler4)
SEH4 frame. previous trylevel=0
SEH4 header:   GSCookieOffset=0xffffffff GSCookieXOROffset=0x0
                EHCookieOffset=0xfffffffcc EHCookieXOROffset=0x0
scopetable entry[0]. previous try level=-2, filter=0x771f74d0 ( ↴
  ↳ ntdll.dll!__safe_se_handler_table+0x20) handler=0x ↴
  ↳ x771f90eb (ntdll.dll!_TppTerminateProcess@4+0x43)
* SEH frame at 0x18ffe4 prev=0xffffffff handler=0x77247428 ( ↴
  ↳ ntdll.dll!_FinalExceptionHandler@16)
```

. ?? _mainCRTStartup(), ..crt/src/winxfltr.c.

ntdll.dll, ntdll.dll, .

: :SEH3 y SEH4.

SEH3:

```
#include <stdio.h>
#include <windows.h>
#include <excpt.h>

int filter_user_exceptions (unsigned int code, struct _EXCEPTION_POINTERS *ep)
{
    printf("in filter. code=0x%08X\n", code);
    if (code == 0x112233)
    {
        printf("yes, that is our exception\n");
        return EXCEPTION_EXECUTE_HANDLER;
    }
    else
    {
        printf("not our exception\n");
        return EXCEPTION_CONTINUE_SEARCH;
    };
}
int main()
{
    int* p = NULL;
    __try
    {
        __try
        {
            printf ("hello!\n");
            RaiseException (0x112233, 0, 0, NULL);
            printf ("0x112233 raised. now let's crash\n");
            *p = 13;      // causes an access violation exception
        ;
        }
        __except(GetExceptionCode() == EXCEPTION_ACCESS_VIOLATION,
        ?
            EXCEPTION_EXECUTE_HANDLER : EXCEPTION_CONTINUE_SEARCH)
        {
            printf("access violation, can't recover\n");
        }
    }
}
```

```

__except(filter_user_exceptions(GetExceptionCode(), ↴
    ↴ GetExceptionInformation()))
{
    // the filter_user_exceptions() function answering to ↴
    // the question
    // "is this exception belongs to this block?"
    // if yes, do the follow:
    printf("user exception caught\n");
}
}
}

```

. scope table . Previous try level .

Listing 68.8: MSVC 2003

```

$SG74606 DB      'in filter. code=0x%08X', 0aH, 00H
$SG74608 DB      'yes, that is our exception', 0aH, 00H
$SG74610 DB      'not our exception', 0aH, 00H
$SG74617 DB      'hello!', 0aH, 00H
$SG74619 DB      '0x112233 raised. now let's crash', 0aH, 00H
$SG74621 DB      'access violation, can't recover', 0aH, 00H
$SG74623 DB      'user exception caught', 0aH, 00H

_code$ = 8      ; size = 4
_ep$ = 12     ; size = 4
_filter_user_exceptions PROC NEAR
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _code$[ebp]
    push    eax
    push    OFFSET FLAT:$SG74606 ; 'in filter. code=0x%08X'
    call    _printf
    add    esp, 8
    cmp    DWORD PTR _code$[ebp], 1122867; 00112233H
    jne    SHORT $L74607
    push    OFFSET FLAT:$SG74608 ; 'yes, that is our exception'
    call    _printf
    add    esp, 4
    mov    eax, 1
    jmp    SHORT $L74605
$L74607:
    push    OFFSET FLAT:$SG74610 ; 'not our exception'
    call    _printf
    add    esp, 4
    xor    eax, eax
$L74605:
    pop    ebp
    ret    0

```

```

_filter_user_exceptions ENDP

; scope table:
CONST SEGMENT
$T74644 DD 0xffffffffH ; previous try level for outer ↵
    ↳ block
        DD FLAT:$L74634 ; outer block filter
        DD FLAT:$L74635 ; outer block handler
        DD 00H           ; previous try level for inner ↵
    ↳ block
        DD FLAT:$L74638 ; inner block filter
        DD FLAT:$L74639 ; inner block handler
CONST ENDS

$T74643 = -36      ; size = 4
$T74642 = -32      ; size = 4
_p$ = -28          ; size = 4
__$SEHRec$ = -24   ; size = 24
_main PROC NEAR
    push ebp
    mov  ebp, esp
    push -1 ; previous try level
    push OFFSET FLAT:$T74644
    push OFFSET FLAT:_except_handler3
    mov  eax, DWORD PTR fs:_except_list
    push eax
    mov  DWORD PTR fs:_except_list, esp
    add esp, -20
    push ebx
    push esi
    push edi
    mov  DWORD PTR __$SEHRec$[ebp], esp
    mov  DWORD PTR _p$[ebp], 0
    mov  DWORD PTR __$SEHRec$[ebp+20], 0 ; outer try block ↵
    ↳ entered. set previous try level to 0
    mov  DWORD PTR __$SEHRec$[ebp+20], 1 ; inner try block ↵
    ↳ entered. set previous try level to 1
    push OFFSET FLAT:$SG74617 ; 'hello!'
    call _printf
    add esp, 4
    push 0
    push 0
    push 0
    push 1122867 ; 00112233H
    call DWORD PTR __imp__RaiseException@16
    push OFFSET FLAT:$SG74619 ; '0x112233 raised. now let's ↵
    ↳ crash'

```

```

call  _printf
add   esp, 4
mov   eax, DWORD PTR _p$[ebp]
mov   DWORD PTR [eax], 13
mov   DWORD PTR __$SEHRec$[ebp+20], 0 ; inner try block ↵
↳ exited. set previous try level back to 0
jmp   SHORT $L74615

; inner block filter:
$L74638:
$L74650:
    mov   ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov   edx, DWORD PTR [ecx]
    mov   eax, DWORD PTR [edx]
    mov   DWORD PTR $T74643[ebp], eax
    mov   eax, DWORD PTR $T74643[ebp]
    sub   eax, -1073741819; c0000005H
    neg   eax
    sbb   eax, eax
    inc   eax
$L74640:
$L74648:
    ret   0

; inner block handler:
$L74639:
    mov   esp, DWORD PTR __$SEHRec$[ebp]
    push  OFFSET FLAT:$SG74621 ; 'access violation, can't ↵
    ↳ recover'
    call  _printf
    add   esp, 4
    mov   DWORD PTR __$SEHRec$[ebp+20], 0 ; inner try block ↵
    ↳ exited. set previous try level back to 0

$L74615:
    mov   DWORD PTR __$SEHRec$[ebp+20], -1 ; outer try block ↵
    ↳ exited, set previous try level back to -1
    jmp   SHORT $L74633

; outer block filter:
$L74634:
$L74651:
    mov   ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov   edx, DWORD PTR [ecx]
    mov   eax, DWORD PTR [edx]
    mov   DWORD PTR $T74642[ebp], eax
    mov   ecx, DWORD PTR __$SEHRec$[ebp+4]

```

```

push    ecx
mov     edx, DWORD PTR $T74642[ebp]
push    edx
call    _filter_user_exceptions
add    esp, 8
$L74636:
$L74649:
ret    0

; outer block handler:
$L74635:
mov     esp, DWORD PTR __$SEHRec$[ebp]
push    OFFSET FLAT:$SG74623 ; 'user exception caught'
call    _printf
add    esp, 4
mov     DWORD PTR __$SEHRec$[ebp+20], -1 ; both try blocks ↴
↳ exited. set previous try level back to -1
$L74633:
xor    eax, eax
mov     ecx, DWORD PTR __$SEHRec$[ebp+8]
mov     DWORD PTR fs:_except_list, ecx
pop    edi
pop    esi
pop    ebx
mov     esp, ebp
pop    ebp
ret    0
_main   ENDP

```

printf().. scope table .

```
tracer.exe -l:3.exe bpx=3.exe!printf --dump-seh
```

Listing 68.9: tracer.exe output

```

(0) 3.exe!printf
EAX=0x00000001b EBX=0x00000000 ECX=0x0040cc58 EDX=0x0008e3c8
ESI=0x00000000 EDI=0x00000000 EBP=0x0018f840 ESP=0x0018f838
EIP=0x004011b6
FLAGS=PF ZF IF
* SEH frame at 0x18f88c prev=0x18fe9c handler=0x771db4ad (ntdll.dll!ExecuteHandler2@20+0x3a)
* SEH frame at 0x18fe9c prev=0x18ff78 handler=0x4012e0 (3.exe!_except_handler3)
SEH3 frame. previous trylevel=1
scopetable entry[0]. previous try level=-1, filter=0x401120 (3.exe!main+0xb0) handler=0x40113b (3.exe!main+0xcb)

```

```

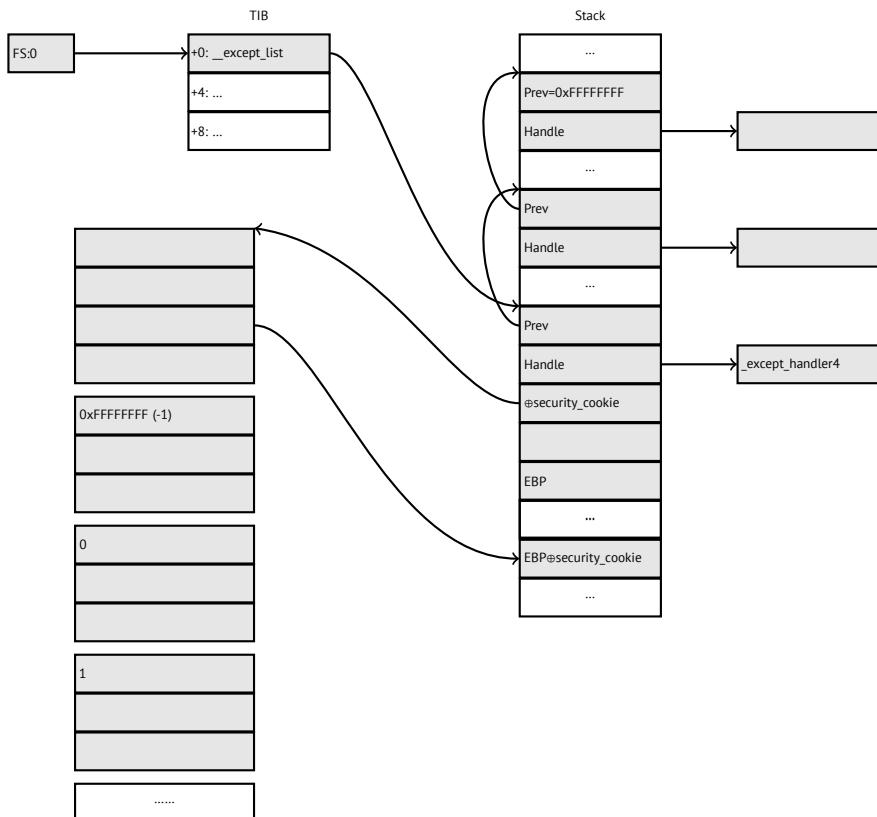
scopetable entry[1]. previous try level=0, filter=0x4010e8 (3. ↴
    ↴ exe!main+0x78) handler=0x401100 (3.exe!main+0x90)
* SEH frame at 0x18ff78 prev=0x18ffc4 handler=0x4012e0 (3.exe! ↴
    ↴ _except_handler3)
SEH3 frame. previous trylevel=0
scopetable entry[0]. previous try level=-1, filter=0x40160d (3. ↴
    ↴ exe!mainCRTStartup+0x18d) handler=0x401621 (3.exe! ↴
    ↴ mainCRTStartup+0xa1)
* SEH frame at 0x18ffc4 prev=0x18ffe4 handler=0x771f71f5 (ntdll, ↴
    ↴ .dll!_except_handler4)
SEH4 frame. previous trylevel=0
SEH4 header:   GSCookieOffset=0xfffffffffe GSCookieXOROffset=0x0
                EHCookieOffset=0xffffffffcc EHCookieXOROffset=0x0
scopetable entry[0]. previous try level=-2, filter=0x771f74d0 ( ↴
    ↴ ntdll.dll!__safe_se_handler_table+0x20) handler=0x ↴
    ↴ x771f90eb (ntdll.dll!_TppTerminateProcess@4+0x43)
* SEH frame at 0x18ffe4 prev=0xffffffff handler=0x77247428 ( ↴
    ↴ ntdll.dll!_FinalExceptionHandler@16)

```

SEH4

([18.2 on page 334](#)) *scope table* MSVC 2005, SEH3 . *scope table* [security cookie](#).
security cookies. [security_cookie](#).

previous try level -2 en SEH4 -1.



MSVC 2012 SEH4:

Listing 68.10: MSVC 2012: one try block example

```
$SG85485 DB      'hello #1!', 0aH, 00H
$SG85486 DB      'hello #2!', 0aH, 00H
$SG85488 DB      'access violation, can''t recover', 0aH, 00H

; scope table:
xdata$x      SEGMENT
__sehtable$_main DD 0ffffffeH    ; GS Cookie Offset
                  DD 00H          ; GS Cookie XOR Offset
                  DD 0fffffcch     ; EH Cookie Offset
                  DD 00H          ; EH Cookie XOR Offset
                  DD 0ffffffeH    ; previous try level
                  DD FLAT:$LN12@main ; filter
                  DD FLAT:$LN8@main  ; handler
xdata$x      ENDS
```

```

$T2 = -36           ; size = 4
_p$ = -32          ; size = 4
_tv68 = -28         ; size = 4
__$SEHRec$ = -24   ; size = 24
_main    PROC
    push    ebp
    mov     ebp, esp
    push    -2
    push    OFFSET __sehtable$_main
    push    OFFSET __except_handler4
    mov     eax, DWORD PTR fs:0
    push    eax
    add    esp, -20
    push    ebx
    push    esi
    push    edi
    mov     eax, DWORD PTR __security_cookie
    xor     DWORD PTR __$SEHRec$[ebp+16], eax ; xored pointer to ↴
    ↴ scope table
    xor     eax, ebp
    push    eax                               ; ebp ^ ↴
    ↴ security_cookie
    lea     eax, DWORD PTR __$SEHRec$[ebp+8] ; pointer to ↴
    ↴ VC_EXCEPTION_REGISTRATION_RECORD
    mov     DWORD PTR fs:0, eax
    mov     DWORD PTR __$SEHRec$[ebp], esp
    mov     DWORD PTR _p$[ebp], 0
    mov     DWORD PTR __$SEHRec$[ebp+20], 0 ; previous try level
    push    OFFSET $SG85485 ; 'hello #1!'
    call    _printf
    add    esp, 4
    mov     eax, DWORD PTR _p$[ebp]
    mov     DWORD PTR [eax], 13
    push    OFFSET $SG85486 ; 'hello #2!'
    call    _printf
    add    esp, 4
    mov     DWORD PTR __$SEHRec$[ebp+20], -2 ; previous try ↴
    ↴ level
    jmp    SHORT $LN6@main

; filter:
$LN7@main:
$LN12@main:
    mov     ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov     edx, DWORD PTR [ecx]
    mov     eax, DWORD PTR [edx]
    mov     DWORD PTR $T2[ebp], eax

```

```

    cmp      DWORD PTR $T2[ebp], -1073741819 ; c0000005H
    jne      SHORT $LN4@main
    mov      DWORD PTR tv68[ebp], 1
    jmp      SHORT $LN5@main
$LN4@main:
    mov      DWORD PTR tv68[ebp], 0
$LN5@main:
    mov      eax, DWORD PTR tv68[ebp]
$LN9@main:
$LN11@main:
    ret      0

; handler:
$LN8@main:
    mov      esp, DWORD PTR __$SEHRec$[ebp]
    push     OFFSET $SG85488 ; 'access violation, can''t recover'
    call     _printf
    add     esp, 4
    mov      DWORD PTR __$SEHRec$[ebp+20], -2 ; previous try ↴
    ↴ level
$LN6@main:
    xor      eax, eax
    mov      ecx, DWORD PTR __$SEHRec$[ebp+8]
    mov      DWORD PTR fs:0, ecx
    pop     ecx
    pop     edi
    pop     esi
    pop     ebx
    mov      esp, ebp
    pop     ebp
    ret      0
_main    ENDP

```

Listing 68.11: MSVC 2012: two try blocks example

```

$SG85486 DB      'in filter. code=0x%08X', 0aH, 00H
$SG85488 DB      'yes, that is our exception', 0aH, 00H
$SG85490 DB      'not our exception', 0aH, 00H
$SG85497 DB      'hello!', 0aH, 00H
$SG85499 DB      '0x112233 raised. now let''s crash', 0aH, 00H
$SG85501 DB      'access violation, can''t recover', 0aH, 00H
$SG85503 DB      'user exception caught', 0aH, 00H

xdata$x      SEGMENT
__sehtable$_main DD 0fffffffFeH           ; GS Cookie Offset
                  DD 00H                 ; GS Cookie XOR Offset
                  DD 0fffffc8H           ; EH Cookie Offset
                  DD 00H                 ; EH Cookie Offset

```

```

        DD      0xffffffffH    ; previous try level for ↴
↳ outer block
        DD      FLAT:$LN19@main ; outer block filter
        DD      FLAT:$LN9@main  ; outer block handler
        DD      00H              ; previous try level for ↴
↳ inner block
        DD      FLAT:$LN18@main ; inner block filter
        DD      FLAT:$LN13@main ; inner block handler
xdata$x    ENDS

$T2 = -40           ; size = 4
$T3 = -36           ; size = 4
_p$ = -32           ; size = 4
tv72 = -28          ; size = 4
__$SEHRec$ = -24   ; size = 24
_main    PROC
    push    ebp
    mov     ebp, esp
    push    -2    ; initial previous try level
    push    OFFSET __sehtable$_main
    push    OFFSET __except_handler4
    mov     eax, DWORD PTR fs:0
    push    eax ; prev
    add    esp, -24
    push    ebx
    push    esi
    push    edi
    mov     eax, DWORD PTR __security_cookie
    xor     DWORD PTR __$SEHRec$[ebp+16], eax      ; xored ↴
↳ pointer to scope table
    xor     eax, ebp                                ; ebp ^ ↴
↳ security_cookie
    push    eax
    lea     eax, DWORD PTR __$SEHRec$[ebp+8]       ; pointer to ↴
↳ VC_EXCEPTION_REGISTRATION_RECORD
    mov     DWORD PTR fs:0, eax
    mov     DWORD PTR __$SEHRec$[ebp], esp
    mov     DWORD PTR _p$[ebp], 0
    mov     DWORD PTR __$SEHRec$[ebp+20], 0 ; entering outer try ↴
↳ block, setting previous try level=0
    mov     DWORD PTR __$SEHRec$[ebp+20], 1 ; entering inner try ↴
↳ block, setting previous try level=1
    push    OFFSET $SG85497 ; 'hello!'
    call    _printf
    add    esp, 4
    push    0
    push    0

```

```

push    0
push    1122867 ; 00112233H
call    DWORD PTR __imp__RaiseException@16
push    OFFSET $SG85499 ; '0x112233 raised. now let's crash'
        \
call    _printf
add    esp, 4
mov    eax, DWORD PTR _p$[ebp]
mov    DWORD PTR [eax], 13
mov    DWORD PTR __$SEHRec$[ebp+20], 0 ; exiting inner try ↴
        ↴ block, set previous try level back to 0
jmp    SHORT $LN2@main

; inner block filter:
$LN12@main:
$LN18@main:
    mov    ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov    edx, DWORD PTR [ecx]
    mov    eax, DWORD PTR [edx]
    mov    DWORD PTR $T3[ebp], eax
    cmp    DWORD PTR $T3[ebp], -1073741819 ; c0000005H
    jne    SHORT $LN5@main
    mov    DWORD PTR tv72[ebp], 1
    jmp    SHORT $LN6@main
$LN5@main:
    mov    DWORD PTR tv72[ebp], 0
$LN6@main:
    mov    eax, DWORD PTR tv72[ebp]
$LN14@main:
$LN16@main:
    ret    0

; inner block handler:
$LN13@main:
    mov    esp, DWORD PTR __$SEHRec$[ebp]
    push   OFFSET $SG85501 ; 'access violation, can't recover'
    call    _printf
    add    esp, 4
    mov    DWORD PTR __$SEHRec$[ebp+20], 0 ; exiting inner try ↴
        ↴ block, setting previous try level back to 0
$LN2@main:
    mov    DWORD PTR __$SEHRec$[ebp+20], -2 ; exiting both ↴
        ↴ blocks, setting previous try level back to -2
    jmp    SHORT $LN7@main

; outer block filter:
$LN8@main:

```

```

$LN19@main:
    mov    ecx, DWORD PTR __$SEHRec$[ebp+4]
    mov    edx, DWORD PTR [ecx]
    mov    eax, DWORD PTR [edx]
    mov    DWORD PTR $T2[ebp], eax
    mov    ecx, DWORD PTR __$SEHRec$[ebp+4]
    push   ecx
    mov    edx, DWORD PTR $T2[ebp]
    push   edx
    call   _filter_user_exceptions
    add    esp, 8

$LN10@main:
$LN17@main:
    ret    0

; outer block handler:
$LN9@main:
    mov    esp, DWORD PTR __$SEHRec$[ebp]
    push   OFFSET $SG85503 ; 'user exception caught'
    call   _printf
    add    esp, 4
    mov    DWORD PTR __$SEHRec$[ebp+20], -2 ; exiting both ↴
    ↴ blocks, setting previous try level back to -2

$LN7@main:
    xor    eax, eax
    mov    ecx, DWORD PTR __$SEHRec$[ebp+8]
    mov    DWORD PTR fs:0, ecx
    pop    ecx
    pop    edi
    pop    esi
    pop    ebx
    mov    esp, ebp
    pop    ebp
    ret    0

_main    ENDP

_code$ = 8    ; size = 4
_ep$ = 12   ; size = 4
_filter_user_exceptions PROC
    push   ebp
    mov    ebp, esp
    mov    eax, DWORD PTR _code$[ebp]
    push   eax
    push   OFFSET $SG85486 ; 'in filter. code=0x%08X'
    call   _printf
    add    esp, 8
    cmp    DWORD PTR _code$[ebp], 1122867 ; 00112233H

```

```

jne    SHORT $LN2@filter_use
push   OFFSET $SG85488 ; 'yes, that is our exception'
call   _printf
add    esp, 4
mov    eax, 1
jmp    SHORT $LN3@filter_use
jmp    SHORT $LN3@filter_use
$LN2@filter_use:
push   OFFSET $SG85490 ; 'not our exception'
call   _printf
add    esp, 4
xor    eax, eax
$LN3@filter_use:
pop    ebp
ret    0
_filter_user_exceptions ENDP

```

cookies:Cookie Offset EBP⊕security_cookie.Cookie XOR Offset EBP⊕security_cookie . :

$$\text{security_cookie} \oplus (\text{CookieXOROffset} + \text{address_of_saved_EBP}) == \\ \text{stack}[\text{address_of_saved_EBP} + \text{CookieOffset}]$$

Cookie Offset -2,.

[tracer](#), [GitHub](#) .

68.3.3 Windows x64

. previous try level . .pdata, .

x64:

Listing 68.12: MSVC 2012

```

$SG86276 DB      'hello #1!', 0aH, 00H
$SG86277 DB      'hello #2!', 0aH, 00H
$SG86279 DB      'access violation, can''t recover', 0aH, 00H

pdata  SEGMENT
$pdata$main DD  imagerel $LN9
          DD  imagerel $LN9+61
          DD  imagerel $unwind$main
pdata  ENDS
pdata  SEGMENT
$pdata$main$filt$0 DD imagerel main$filt$0

```

```

        DD      imagerel main$filter$0+32
        DD      imagerel $unwind$main$filter$0
pdata    ENDS
xdata   SEGMENT
$unwind$main DD 020609H
        DD      030023206H
        DD      imagerel __C_specific_handler
        DD      01H
        DD      imagerel $LN9+8
        DD      imagerel $LN9+40
        DD      imagerel main$filter$0
        DD      imagerel $LN9+40
$unwind$main$filter$0 DD 020601H
        DD      050023206H
xdata   ENDS

_TEXT  SEGMENT
main   PROC
$LN9:
    push   rbx
    sub    rsp, 32
    xor    ebx, ebx
    lea    rcx, OFFSET FLAT:$SG86276 ; 'hello #1!'
    call   printf
    mov    DWORD PTR [rbx], 13
    lea    rcx, OFFSET FLAT:$SG86277 ; 'hello #2!'
    call   printf
    jmp    SHORT $LN8@main
$LN6@main:
    lea    rcx, OFFSET FLAT:$SG86279 ; 'access violation, ↴
    ↴ can't recover'
    call   printf
    npad  1 ; align next label
$LN8@main:
    xor    eax, eax
    add    rsp, 32
    pop    rbp
    ret    0
main   ENDP
_TEXT  ENDS

text$x  SEGMENT
main$filter$0 PROC
    push   rbp
    sub    rsp, 32
    mov    rbp, rdx
$LN5@main$filter$0:

```

```

        mov      rax, QWORD PTR [rcx]
        xor      ecx, ecx
        cmp      DWORD PTR [rax], -1073741819; c0000005H
        sete    cl
        mov      eax, ecx
$LN7@main$filter$:
        add      rsp, 32
        pop      rbp
        ret      0
        int      3
main$filter$0 ENDP
text$x  ENDS

```

Listing 68.13: MSVC 2012

```

$SG86277 DB      'in filter. code=0x%08X', 0aH, 00H
$SG86279 DB      'yes, that is our exception', 0aH, 00H
$SG86281 DB      'not our exception', 0aH, 00H
$SG86288 DB      'hello!', 0aH, 00H
$SG86290 DB      '0x112233 raised. now let's crash', 0aH, 00H
$SG86292 DB      'access violation, can't recover', 0aH, 00H
$SG86294 DB      'user exception caught', 0aH, 00H

pdata   SEGMENT
$pdata$filter_user_exceptions DD imagerel $LN6
        DD      imagerel $LN6+73
        DD      imagerel $unwind$filter_user_exceptions
$pdata$main DD      imagerel $LN14
        DD      imagerel $LN14+95
        DD      imagerel $unwind$main
pdata   ENDS
pdata   SEGMENT
$pdata$main$filter$0 DD imagerel main$filter$0
        DD      imagerel main$filter$0+32
        DD      imagerel $unwind$main$filter$0
$pdata$main$filter$1 DD imagerel main$filter$1
        DD      imagerel main$filter$1+30
        DD      imagerel $unwind$main$filter$1
pdata   ENDS

xdata   SEGMENT
$unwind$filter_user_exceptions DD 020601H
        DD      030023206H
$unwind$main DD 020609H
        DD      030023206H
        DD      imagerel __C_specific_handler
        DD      02H
        DD      imagerel $LN14+8

```

```

        DD      imagerel $LN14+59
        DD      imagerel main$filter$0
        DD      imagerel $LN14+59
        DD      imagerel $LN14+8
        DD      imagerel $LN14+74
        DD      imagerel main$filter$1
        DD      imagerel $LN14+74
$unwind$main$filter$0 DD 020601H
        DD      050023206H
$unwind$main$filter$1 DD 020601H
        DD      050023206H
xdata ENDS

_TEXT SEGMENT
main PROC
$LN14:
    push    rbx
    sub     rsp, 32
    xor     ebx, ebx
    lea     rcx, OFFSET FLAT:$SG86288 ; 'hello!'
    call    printf
    xor     r9d, r9d
    xor     r8d, r8d
    xor     edx, edx
    mov     ecx, 1122867 ; 00112233H
    call    QWORD PTR __imp_RaiseException
    lea     rcx, OFFSET FLAT:$SG86290 ; '0x112233 raised. ↴
    ↳ now let's crash'
    call    printf
    mov     DWORD PTR [rbx], 13
    jmp    SHORT $LN13@main
$LN11@main:
    lea     rcx, OFFSET FLAT:$SG86292 ; 'access violation, ↴
    ↳ can't recover'
    call    printf
    npad   1 ; align next label
$LN13@main:
    jmp    SHORT $LN9@main
$LN7@main:
    lea     rcx, OFFSET FLAT:$SG86294 ; 'user exception ↴
    ↳ caught'
    call    printf
    npad   1 ; align next label
$LN9@main:
    xor     eax, eax
    add     rsp, 32
    pop    rbx

```

```

        ret    0
main    ENDP

text$x SEGMENT
main$filter$0 PROC
    push    rbp
    sub     rsp, 32
    mov     rbp, rdx
$LN10@main$filter$:
    mov     rax, QWORD PTR [rcx]
    xor     ecx, ecx
    cmp     DWORD PTR [rax], -1073741819; c0000005H
    sete   cl
    mov     eax, ecx
$LN12@main$filter$:
    add     rsp, 32
    pop    rbp
    ret    0
    int    3
main$filter$0 ENDP

main$filter$1 PROC
    push    rbp
    sub     rsp, 32
    mov     rbp, rdx
$LN6@main$filter$:
    mov     rax, QWORD PTR [rcx]
    mov     rdx, rcx
    mov     ecx, DWORD PTR [rax]
    call    filter_user_exceptions
    npad   1 ; align next label
$LN8@main$filter$:
    add     rsp, 32
    pop    rbp
    ret    0
    int    3
main$filter$1 ENDP
text$x ENDS

_TEXT  SEGMENT
code$ = 48
ep$ = 56
filter_user_exceptions PROC
$LN6:
    push    rbx
    sub     rsp, 32
    mov     ebx, ecx

```

```

        mov    edx, ecx
        lea    rcx, OFFSET FLAT:$SG86277 ; 'in filter. code=0x2
        ↴ %08X'
        call   printf
        cmp    ebx, 1122867; 00112233H
        jne    SHORT $LN2@filter_use
        lea    rcx, OFFSET FLAT:$SG86279 ; 'yes, that is our ↴
        ↴ exception'
        call   printf
        mov    eax, 1
        add    rsp, 32
        pop    rbx
        ret    0
$LN2@filter_use:
        lea    rcx, OFFSET FLAT:$SG86281 ; 'not our exception'
        call   printf
        xor    eax, eax
        add    rsp, 32
        pop    rbx
        ret    0
filter_user_exceptions ENDP
_TEXT    ENDS

```

[[Sko12](#)].

.pdata .

68.3.4 SEH

[[Pie](#)], [[Sko12](#)].

68.4 Windows NT:

Las secciones críticas en cualquier OS son muy importantes en un entorno multihilado, principalmente para garantizar que solo un hilo pueda acceder a algunos datos en un momento dado, bloqueando otros hilos e interrupciones.

CRITICAL_SECTION Windows NT:

Listing 68.14: (Windows Research Kernel v1.2) public/sdk/inc/nturtl.h

```

typedef struct _RTL_CRITICAL_SECTION {
    PRTL_CRITICAL_SECTION_DEBUG DebugInfo;
}
//
```

```

// The following three fields control entering and exiting
// the critical
// section for the resource
//

LONG LockCount;
LONG RecursionCount;
HANDLE OwningThread;           // from the thread's ClientId->
// UniqueThread
HANDLE LockSemaphore;
ULONG_PTR SpinCount;          // force size on 64-bit systems
// when packed
} RTL_CRITICAL_SECTION, *PRTL_CRITICAL_SECTION;

```

EnterCriticalSection():

Listing 68.15: Windows 2008/ntdll.dll/x86 (begin)

```
_RtlEnterCriticalSection@4
```

```

var_C      = dword ptr -0Ch
var_8      = dword ptr -8
var_4      = dword ptr -4
arg_0      = dword ptr 8

        mov     edi, edi
        push    ebp
        mov     ebp, esp
        sub     esp, 0Ch
        push    esi
        push    edi
        mov     edi, [ebp+arg_0]
        lea     esi, [edi+4] ; LockCount
        mov     eax, esi
        lock btr dword ptr [eax], 0
        jnb     wait ; jump if CF=0

```

```
loc_7DE922DD:
```

```

        mov     eax, large fs:18h
        mov     ecx, [eax+24h]
        mov     [edi+0Ch], ecx
        mov     dword ptr [edi+8], 1
        pop     edi
        xor     eax, eax
        pop     esi
        mov     esp, ebp
        pop     ebp
        retn   4

```

... skipped

BTR (LOCK): .., LockCount 1, ..
WaitForSingleObject().

LeaveCriticalSection():

Listing 68.16: Windows 2008/ntdll.dll/x86 (begin)

```
_RtlLeaveCriticalSection@4 proc near

arg_0          = dword ptr  8

        mov     edi, edi
        push    ebp
        mov     ebp, esp
        push    esi
        mov     esi, [ebp+arg_0]
        add     dword ptr [esi+8], 0FFFFFFFh ; ↴
        ↳ RecursionCount
        jnz     short loc_7DE922B2
        push    ebx
        push    edi
        lea     edi, [esi+4]      ; LockCount
        mov     dword ptr [esi+0Ch], 0
        mov     ebx, 1
        mov     eax, edi
        lock   xadd [eax], ebx
        inc     ebx
        cmp     ebx, 0FFFFFFFh
        jnz     loc_7DEA8EB7

loc_7DE922B0:
        pop     edi
        pop     ebx

loc_7DE922B2:
        xor     eax, eax
        pop     esi
        pop     ebp
        retn   4

... skipped
```

XADD...

LOCK :

Parte VII

Herramientas

Capítulo 69

Desensamblador

69.1 IDA

Una versión gratuita más antigua está disponible para descarga ¹.

Cheatsheet de teclas de acceso rápido: [F.1 on page 1255](#)

¹hex-rays.com/products/ida/support/download_freeware.shtml

Capítulo 70

Depurador

70.1 OllyDbg

Depurador de modo usuario win32 muy popular:
ollydbg.de.

Cheatsheet de teclas de acceso rápido: [F.2 on page 1256](#)

70.2 GDB

Depurador no muy popular entre ingenieros inversos, pero muy cómodo de todos modos. Algunos comandos: [F.5 on page 1257](#).

70.3 tracer

El autor a menudo usa *tracer*¹ en lugar de un depurador.

El autor de estas líneas dejó de usar un depurador eventualmente, ya que todo lo que necesita de él es detectar argumentos de función mientras se ejecuta, o el estado de los registros en algún punto. Cargar un depurador cada vez es demasiado, así que nació una pequeña utilidad llamada *tracer*. Funciona desde la línea de comandos, permite interceptar la ejecución de funciones, establecer puntos de interrupción en lugares arbitrarios, leer y cambiar el estado de los registros, etc.

Sin embargo, para fines de aprendizaje es altamente recomendable trazar código en un depurador manualmente, observar cómo cambia el estado de los registros

1

(ej. SoftICE clásico, OllyDbg, WinDbg resaltan registros cambiados), flags, datos, cambiarlos manualmente, observar la reacción, etc.

Capítulo 71

Rastreo de llamadas al sistema

71.0.1 strace / dtruss

Muestra qué llamadas al sistema ([syscalls\(66 on page 886\)](#)) están siendo llamadas por un proceso en este momento. Por ejemplo:

En Mac OS X hay dtruss para hacer lo mismo.

Cygwin también tiene strace, pero hasta donde se sabe, funciona solo para archivos .exe compilados para el propio entorno cygwin.

Capítulo 72

Descompiladores

Hay solo uno conocido, públicamente disponible, descompilador de alta calidad para código C: Hex-Rays:

hex-rays.com/products/decompiler/

Capítulo 73

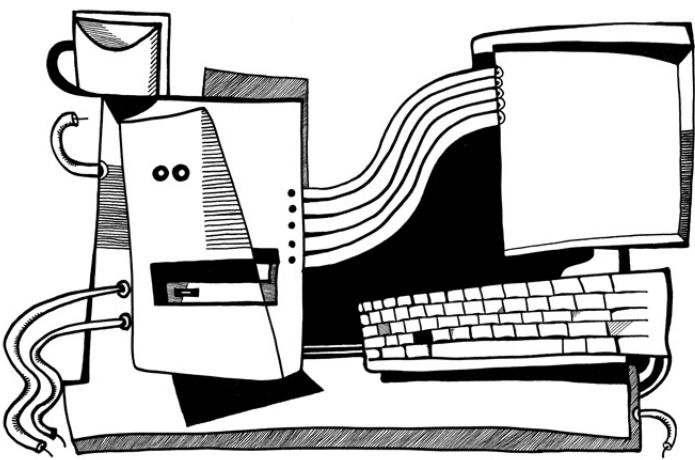
Otras herramientas

- Microsoft Visual Studio Express¹: Versión gratuita reducida de Visual Studio, conveniente para experimentos simples. Algunas opciones útiles: [F.3 on page 1257](#).
- Hiew² para pequeñas modificaciones de código en archivos binarios.
- binary grep: una pequeña utilidad para buscar cualquier secuencia de bytes en una gran pila de archivos, incluyendo no ejecutables: [GitHub](#).

¹visualstudio.com/en-US/products/visual-studio-express-vs

²hiew.ru

Parte VIII



Capítulo 74

(Windows Vista)

taskmgr.exe.¹).

taskmgr.exe en IDA. CAdapter, CNetPage, CPerfPage, CProcInfo, CProcPage, CSvcPage, CTaskPage, CUserPage.

xrefs to __imp_NtQuerySystemInformation		
Dir...	T...	Address
L4 Up	p	wWinMain+50E
L4 Up	p	wWinMain+542
L4 Up	p	CPerfPage::TimeEvent(void)+200
L4 Up	p	InitPerfInfo(void)+2C
L4 D...	p	InitPerfInfo(void)+F0
L4 D...	p	CalcCpuTime(int)+5F
L4 D...	p	CalcCpuTime(int)+248
L4 D...	p	CPerfPage::CalcPhysicalMem(unsigned ...)
L4 D...	p	CPerfPage::CalcPhysicalMem(unsigned ...)
L4 D...	p	CProcPage::GetProcessInfo(void)+2B
L4 D...	p	CProcPage::UpdateProcInfoArray(void)+...
L4 D...	p	CProcPage::Initialize(fwND_...)+201
L4 D...	p	CProcPage::GetTaskListEx(void)+3C

Figura 74.1: IDA: NtQuerySystemInformation()

Listing 74.1: taskmgr.exe (Windows Vista)

.text:10000B4B3	xor	r9d, r9d
.text:10000B4B6	lea	rdx, [rsp+0C78h+var_C58] ; ↴ ↳ buffer
.text:10000B4BB	xor	ecx, ecx
.text:10000B4BD	lea	ebp, [r9+40h]
.text:10000B4C1	mov	r8d, ebp

¹MSDN

```

.text:10000B4C4          call    cs:_
    ↳ __imp_NtQuerySystemInformation ; 0
.text:10000B4CA          xor     ebx, ebx
.text:10000B4CC          cmp     eax, ebx
.text:10000B4CE          jge    short loc_10000B4D7
.text:10000B4D0
.text:10000B4D0 loc_10000B4D0:                                ; CODE 
    ↳ XREF: InitPerfInfo(void)+97
.text:10000B4D0
    ↳ InitPerfInfo(void)+AF
; -----
.text:10000B4D0          xor     al, al
.text:10000B4D2          jmp     loc_10000B5EA
.text:10000B4D7 ; 
    ↳ -----
    ↳
.text:10000B4D7
.text:10000B4D7 loc_10000B4D7:                                ; CODE 
    ↳ XREF: InitPerfInfo(void)+36
.text:10000B4D7          mov     eax, [rsp+0C78h+var_C50]
.text:10000B4DB          mov     esi, ebx
.text:10000B4DD          mov     r12d, 3E80h
.text:10000B4E3          mov     cs:?g_PageSize@@3KA, eax ; 
    ↳ ulong g_PageSize
.text:10000B4E9          shr     eax, 0Ah
.text:10000B4EC          lea     r13, __ImageBase
.text:10000B4F3          imul   eax, [rsp+0C78h+var_C4C]
.text:10000B4F8          cmp     [rsp+0C78h+var_C20], bpl
.text:10000B4FD          mov     cs:?g_MEMMax@@3_JA, rax ; 
    ↳ __int64 g_MEMMax
.text:10000B504          movzx  eax, [rsp+0C78h+var_C20] ; 
    ↳ number of CPUs
.text:10000B509          cmova eax, ebp
.text:10000B50C          cmp     al, bl
.text:10000B50E          mov     cs:?g_cProcessors@@3EA, al ; 
    ↳ uchar g_cProcessors

```

g_cProcessors .

var_C20. var_C58 NtQuerySystemInformation(). 0xC20 y 0xC58 0x38 (56).

```

typedef struct _SYSTEM_BASIC_INFORMATION {
    BYTE Reserved1[24];
    PVOID Reserved2[4];
    CCHAR NumberOfProcessors;
} SYSTEM_BASIC_INFORMATION;

```

. taskmgr.exe C:\Windows\System32 (*Windows Resource Protection*) taskmgr.exe

```

01 0000B4F8: 40386C2458      cmp    [rsp][058],bp
01 0000B4FD: 48890544A00100  mov    [00000001`00025548],rax
01 0000B504: 0FB6442458     movzx eax,b,[rsp][058]
01 0000B509: 0F47C5          cmova eax,ebp
01 0000B50C: 3AC3           cmp    al,b1
01 0000B50E: 880574950100   mov    [00000001`00024A88],al
01 0000B514: 7645           jbe   .00000001`0000B55B --3
01 0000B516: 488FB          mov    rdi,rbx
01 0000B519: 498BD4         mov    rdx,r12
01 0000B51C: 8BCD           mov    ecx,ebp

```

```

00 0000A8F8: 40386C2458      cmp    [rsp][058],bp
00 0000A8FD: 48890544A00100  mov    [000024948],rax
00 0000A904: 66B84000         mov    ax,00040 ; '@'
00 0000A908: 90              nop
00 0000A909: 0F47C5          cmova eax,ebp
00 0000A90C: 3AC3           cmp    al,b1
00 0000A90E: 880574950100   mov    [000023E88],al
00 0000A914: 7645           jbe   00000A95B
00 0000A916: 488FB          mov    rdi,rbx
00 0000A919: 498BD4         mov    rdx,r12
00 0000A91C: 8BCD           mov    ecx,ebp

```

Figura 74.2: Hiew:

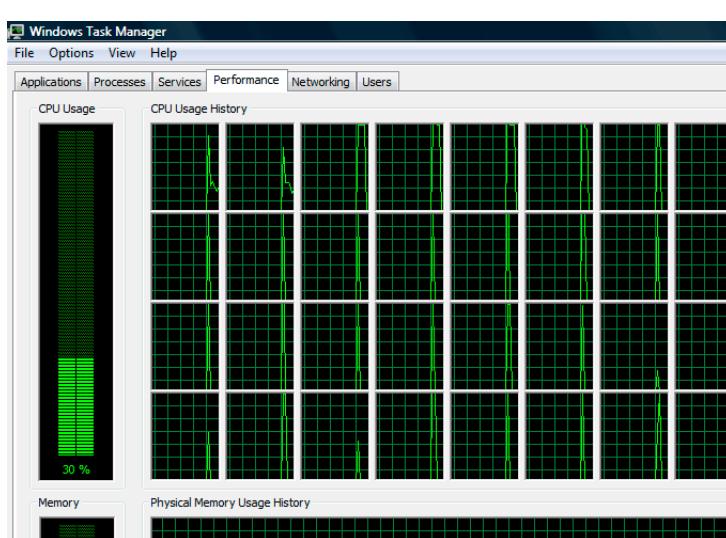


Figura 74.4: Windows Task Manager

74.1

Listing 74.2: taskmgr.exe (Windows Vista)

```

xor      r9d, r9d
div      dword ptr [rsp+4C8h+WndClass.✓
↪ lpfnWndProc]
lea      rdx, [rsp+4C8h+VersionInformation]
lea      ecx, [r9+2]      ; put 2 to ECX
mov      r8d, 138h
mov      ebx, eax
; ECX=SystemPerformanceInformation
call    cs:_imp_NtQuerySystemInformation ; 2
...
mov      r8d, 30h
lea      r9, [rsp+298h+var_268]
lea      rdx, [rsp+298h+var_258]
lea      ecx, [r8-2Dh]    ; put 3 to ECX
; ECX=SystemTimeOfDayInformation
call    cs:_imp_NtQuerySystemInformation ; ✓
↪ not zero
...
mov      rbp, [rsi+8]
mov      r8d, 20h
lea      r9, [rsp+98h+arg_0]
lea      rdx, [rsp+98h+var_78]
lea      ecx, [r8+2Fh]    ; put 0x4F to ECX
mov      [rsp+98h+var_60], ebx
mov      [rsp+98h+var_68], rbp
; ECX=SystemSuperfetchInformation
call    cs:_imp_NtQuerySystemInformation ; ✓
↪ not zero

```

: 64.5.1 on page 874.

Capítulo 75

- . BallTriX, 1997, [http://go.yurichev.com/17311.:](http://go.yurichev.com/17311.)

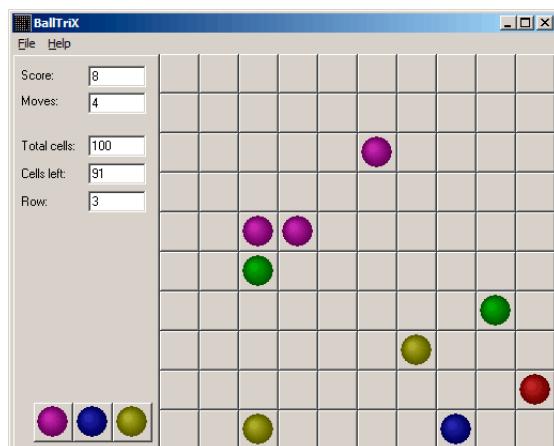


Figura 75.1:

IDA _rand balltrix.exe 0x00403DA0. IDA:

```
.text:00402C9C sub_402C9C      proc near               ; CODE 
    ↳ XREF: sub_402ACA+52
.text:00402C9C
    ↳ sub_402ACA+64 ...
.text:00402C9C
.text:00402C9C arg_0          = dword ptr  8
.text:00402C9C
.text:00402C9C
    push    ebp
.text:00402C9D      mov     ebp, esp
.text:00402C9F      push   ebx
.text:00402CA0      push   esi
.text:00402CA1      push   edi
.text:00402CA2      mov     eax, dword_40D430
.text:00402CA7      imul   eax, dword_40D440
.text:00402CAE      add    eax, dword_40D5C8
.text:00402CB4      mov     ecx, 32000
.text:00402CB9      cdq
.text:00402CBA      idiv   ecx
.text:00402CBC      mov     dword_40D440, edx
.text:00402CC2      call   _rand
.text:00402CC7      cdq
.text:00402CC8      idiv   [ebp+arg_0]
.text:00402CCB      mov     dword_40D430, edx
.text:00402CD1      mov     eax, dword_40D430
.text:00402CD6      jmp    $+5
.text:00402CDB      pop    edi
.text:00402CDC      pop    esi
.text:00402CDD      pop    ebx
.text:00402CDE      leave
.text:00402CDF      retn
.text:00402CDF sub_402C9C      endp
```

«random»..

```
.
:
.
.text:00402B16      mov     eax, dword_40C03C ; 10 
    ↳ here
.text:00402B1B      push   eax
.text:00402B1C      call   random
.text:00402B21      add    esp, 4
.text:00402B24      inc    eax
.text:00402B25      mov    [ebp+var_C], eax
.text:00402B28      mov    eax, dword_40C040 ; 10 
    ↳ here
```

.text:00402B2D	push	eax
.text:00402B2E	call	random
.text:00402B33	add	esp, 4

:

.text:00402BBB	mov	eax, dword_40C058 ; 5 ↵
↳ here		
.text:00402BC0	push	eax
.text:00402BC1	call	random
.text:00402BC6	add	esp, 4
.text:00402BC9	inc	eax

. .! rand() 0..0xFFFF. 0..n - 1 y n ..

. (PUSH/CALL/ADD) NOPs. XOR EAX, EAX.

.00402BB8: 83C410	add	esp, 010
.00402BBB: A158C04000	mov	eax, [00040C058]
.00402BC0: 31C0	xor	eax, eax
.00402BC2: 90	nop	
.00402BC3: 90	nop	
.00402BC4: 90	nop	
.00402BC5: 90	nop	
.00402BC6: 90	nop	
.00402BC7: 90	nop	
.00402BC8: 90	nop	
.00402BC9: 40	inc	eax
.00402BCA: 8B4DF8	mov	ecx, [ebp][-8]
.00402BCD: 8D0C49	lea	ecx, [ecx][ecx]*2
.00402BD0: 8B15F4D54000	mov	edx, [00040D5F4]

random().

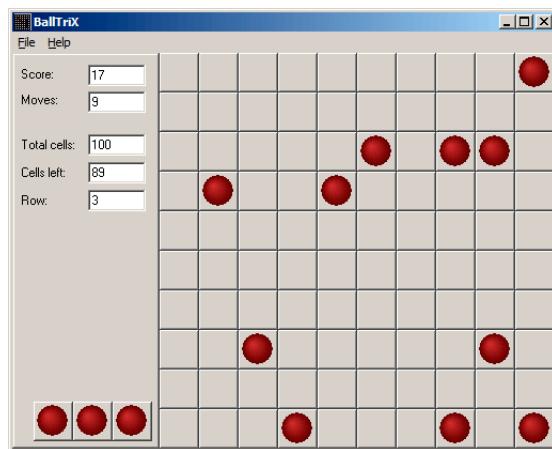


Figura 75.2:

1.

random() ? . 10 y 5 .

Capítulo 76

Buscaminas (Windows XP)

winmine.exe [IDA](#), [PDB](#).

rand():

```
.text:01003940 ; __stdcall Rnd(x)
.text:01003940 _Rnd@4          proc near               ; CODE 
    ↳ XREF: StartGame()+53
.text:01003940                   ; 
    ↳ StartGame()+61
.text:01003940                   ; 
    ↳ StartGame()
.text:01003940 arg_0           = dword ptr  4
.text:01003940                   ; 
.text:01003940                   call     ds:_imp__rand
.text:01003946                   cdq
.text:01003947                   idiv    [esp+arg_0]
.text:0100394B                   mov     eax, edx
.text:0100394D                   retn    4
.text:0100394D _Rnd@4          endp
```

:

```
int Rnd(int limit)
{
    return rand() % limit;
};
```

Rnd() StartGame(),:

.text:010036C7	push	_xBoxMac
.text:010036CD	call	_Rnd@4 ; Rnd(x)
.text:010036D2	push	_yBoxMac
.text:010036D8	mov	esi, eax
.text:010036DA	inc	esi
.text:010036DB	call	_Rnd@4 ; Rnd(x)
.text:010036E0	inc	eax
.text:010036E1	mov	ecx, eax
.text:010036E3	shl	ecx, 5 ; ECX=✓ ↳ ECX*32
.text:010036E6	test	_rgBlk[ecx+esi], 80h
.text:010036EE	jnz	short loc_10036C7
.text:010036F0	shl	eax, 5 ; EAX=✓ ↳ EAX*32
.text:010036F3	lea	eax, _rgBlk[eax+esi]
.text:010036FA	or	byte ptr [eax], 80h
.text:010036FD	dec	_cBombStart
.text:01003703	jnz	short loc_10036C7

Rnd(). OR 0x010036FA. (Rnd()), TEST y JNZ 0x010036E6 .

cBombStart

.

rgBlk rgBlk. 0x360 (864):

.data:01005340 _rgBlk	db 360h dup(?)	; DATA ✓ ↳ XREF: MainWndProc(x,x,x,x)+574
.data:01005340		; ✓ ↳ DisplayBlk(x,x)+23
.data:010056A0 _Preferences	dd ?	; DATA ✓ ↳ XREF: FixMenus()+2
...		

864/32 = 27.

27 * 32? ...

OllyDbg. ¹.

:

Address	Hex dump
01005340	10 10 10 10 10 10 10 10 10 10 10 OF OF OF OF OF
01005350	OF
01005360	10 OF 10 OF OF OF OF OF

¹Windows XP SP3 English. .

01005370	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005380	10	OF	OF	OF		OF	OF	OF		OF	OF	10	OF		OF	OF	OF
01005390	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010053A0	10	OF	OF	OF		OF	OF	OF		8F	OF	10	OF		OF	OF	OF
010053B0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010053C0	10	OF	OF	OF		OF	OF	OF		OF	OF	10	OF		OF	OF	OF
010053D0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010053E0	10	OF	OF	OF		OF	OF	OF		OF	OF	10	OF		OF	OF	OF
010053F0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005400	10	OF	OF	8F		OF	OF	8F		OF	OF	10	OF		OF	OF	OF
01005410	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005420	10	8F	OF	OF		8F	OF	OF		OF	OF	10	OF		OF	OF	OF
01005430	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005440	10	8F	OF	OF		OF	OF	8F		OF	OF	8F	10	OF		OF	OF
01005450	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005460	10	OF	OF	OF		OF	8F	OF		OF	OF	8F	10	OF		OF	OF
01005470	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
01005480	10	10	10	10		10	10	10		10	10	10	10	OF		OF	OF
01005490	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010054A0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010054B0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF
010054C0	OF	OF	OF	OF		OF	OF	OF		OF	OF	OF		OF	OF	OF	OF

OllyDbg, . .



Figura 76.1:

```
border:  
01005340 10 10 10 10 10 10 10 10 10 10 10 10 OF OF OF OF  
01005350 OF  
line #1:
```

```

01005360 10 OF OF OF OF OF OF OF OF 10 OF OF OF OF OF
01005370 OF OF
line #2:
01005380 10 OF OF OF OF OF OF OF OF 10 OF OF OF OF OF
01005390 OF OF
line #3:
010053A0 10 OF OF OF OF OF OF[8F]OF 10 OF OF OF OF OF
010053B0 OF OF
line #4:
010053C0 10 OF OF OF OF OF OF OF OF 10 OF OF OF OF OF
010053D0 OF OF
line #5:
010053E0 10 OF OF OF OF OF OF OF OF 10 OF OF OF OF OF
010053F0 OF OF
line #6:
01005400 10 OF OF[8F]OF OF[8F]OF OF OF 10 OF OF OF OF
01005410 OF OF
line #7:
01005420 10[8F]OF OF[8F]OF OF OF OF 10 OF OF OF OF OF
01005430 OF OF
line #8:
01005440 10[8F]OF OF OF OF[8F]OF OF[8F]10 OF OF OF OF OF
01005450 OF OF
line #9:
01005460 10 OF OF OF OF[8F]OF OF OF[8F]10 OF OF OF OF OF
01005470 OF OF
border:
01005480 10 10 10 10 10 10 10 10 10 10 10 OF OF OF OF OF
01005490 OF OF

```

```

OF OF OF OF OF OF OF OF
OF OF OF OF OF OF OF OF
OF OF OF OF OF OF[8F]OF
OF OF OF OF OF OF OF OF
OF OF OF OF OF OF OF OF
OF OF[8F]OF OF[8F]OF OF OF
[8F]OF OF[8F]OF OF OF OF
[8F]OF OF OF OF[8F]OF OF[8F]
OF OF OF OF[8F]OF OF OF[8F]

```

:

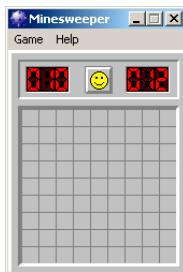


Figura 76.2:

:

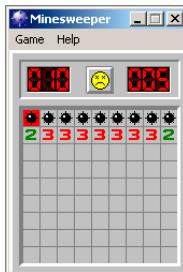


Figura 76.3:

```
// Windows XP MineSweeper cheater
// written by dennis(a)yurichev.com for http://beginners.re/ ↵
    ↵ book
#include <windows.h>
#include <assert.h>
#include <stdio.h>

int main (int argc, char * argv[])
{
    int i, j;
    HANDLE h;
    DWORD PID, address, rd;
    BYTE board[27][32];

    if (argc!=3)
    {
        printf ("Usage: %s <PID> <address>\n", argv[0]) ↵
    ↵ ;
}
```

```
        return 0;
    };

    assert (argv[1]!=NULL);
    assert (argv[2]!=NULL);

    assert (sscanf (argv[1], "%d", &PID)==1);
    assert (sscanf (argv[2], "%x", &address)==1);

    h=OpenProcess (PROCESS_VM_OPERATION | PROCESS_VM_READ | ↴
↳ PROCESS_VM_WRITE, FALSE, PID);

    if (h==NULL)
    {
        DWORD e=GetLastError();
        printf ("OpenProcess error: %08X\n", e);
        return 0;
    };

    if (ReadProcessMemory (h, (LPVOID)address, board, ↴
↳ sizeof(board), &rd)!=TRUE)
    {
        printf ("ReadProcessMemory() failed\n");
        return 0;
    };

    for (i=1; i<26; i++)
    {
        if (board[i][0]==0x10 && board[i][1]==0x10)
            break; // end of board
        for (j=1; j<31; j++)
        {
            if (board[i][j]==0x10)
                break; // board border
            if (board[i][j]==0x8F)
                printf ("*");
            else
                printf (" ");
        };
        printf ("\n");
    };

    CloseHandle (h);
};
```

76.1 Ejercicios

- ?
- .
-
- : [G.5.1 on page 1272](#).

²ID de programa/proceso

³PID Task Manager («View → Select Columns»)

⁴: [beginners.re](#)

Capítulo 77

77.1

IDA:

```
sub_401510    proc near
; ECX = input
    mov     rdx, 5D7E0D1F2E0F1F84h
    mov     rax, rcx          ; input
    imul   rax, rdx
    mov     rdx, 388D76AEE8CB1500h
    mov     ecx, eax
    and    ecx, 0Fh
    ror    rax, cl
    xor    rax, rdx
    mov     rdx, 0D2E9EE7E83C4285Bh
    mov     ecx, eax
    and    ecx, 0Fh
    rol    rax, cl
    lea    r8, [rax+rdx]
    mov     rdx, 8888888888888889h
    mov     rax, r8
    mul    rdx
    shr    rdx, 5
    mov     rax, rdx
    lea    rcx, [r8+rdx*4]
    shl    rax, 6
    sub    rcx, rax
    mov     rax, r8
```

```

        rol    rax, cl
        ; EAX = output
        retn
sub_401510    endp

```

GCC, ECX.

Hex-Rays. :

```

uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;

    ecx=input;

    rdx=0x5D7E0D1F2E0F1F84;
    rax=rcx;
    rax*=rdx;
    rdx=0x388D76AEE8CB1500;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=rdx;
    rdx=0xD2E9EE7E83C4285B;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+rdx;
    rdx=0x8888888888888889;
    rax=r8;
    rax*=rdx;
    rdx=rdx>>5;
    rax=rdx;
    rcx=r8+rdx*4;
    rax=rax<<6;
    rcx=rcx-rax;
    rax=r8
    rax=_lrotl (rax, rcx&0xFF); // rotate left
    return rax;
}

```

!

:

```

uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;

    ecx=input;

    rdx=0x5D7E0D1F2E0F1F84;

```

```
    rax=rcx;
    rax*=rdx;
    rdx=0x388D76AEE8CB1500;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=rdx;
    rdx=0xD2E9EE7E83C4285B;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+rdx;

    rdx=0x8888888888888889;
    rax=r8;
    rax*=rdx;
    // RDX here is a high part of multiplication result
    rdx=rdx>>5;
    // RDX here is division result!
    rax=rdx;

    rcx=r8+rdx*4;
    rax=rax<<6;
    rcx=rcx-rax;
    rax=r8
    rax=_lrotl (rax, rcx&0xFF); // rotate left
    return rax;
};
```

:

```
uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;

    ecx=input;

    rdx=0x5D7E0D1F2E0F1F84;
    rax=rcx;
    rax*=rdx;
    rdx=0x388D76AEE8CB1500;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=rdx;
    rdx=0xD2E9EE7E83C4285B;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+rdx;

    rdx=0x8888888888888889;
    rax=r8;
    rax*=rdx;
    // RDX here is a high part of multiplication result
    rdx=rdx>>5;
```

```
// RDX here is division result!
rax=rdx;

rcx=(r8+rdx*4)-(rax<<6);
rax=r8
rax=_lrotl (rax, rcx&0xFF); // rotate left
return rax;
};
```

(41 on page 631). Wolfram Mathematica:

Listing 77.1: Wolfram Mathematica

```
In[1]:=N[2^(64 + 5)/16^8888888888888889]
Out[1]:=60.
```

```
:
uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;

    ecx=input;

    rdx=0x5D7E0D1F2E0F1F84;
    rax=rcx;
    rax*=rdx;
    rdx=0x388D76AEE8CB1500;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=rdx;
    rdx=0xD2E9EE7E83C4285B;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+rdx;

    rax=rdx=r8/60;

    rcx=(r8+rax*4)-(rax*64);
    rax=r8
    rax=_lrotl (rax, rcx&0xFF); // rotate left
    return rax;
};
```

```
:
uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;
```

```
    rax=input;
    rax*=0x5D7E0D1F2E0F1F84;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=0x388D76AEE8CB1500;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+0xD2E9EE7E83C4285B;

    rcx=r8-(r8/60)*60;
    rax=r8
    rax=_lrotl (rax, rcx&0xFF); // rotate left
    return rax;
};
```

:

```
uint64_t f(uint64_t input)
{
    uint64_t rax, rbx, rcx, rdx, r8;

    rax=input;
    rax*=0x5D7E0D1F2E0F1F84;
    rax=_lrotr(rax, rax&0xF); // rotate right
    rax^=0x388D76AEE8CB1500;
    rax=_lrotl(rax, rax&0xF); // rotate left
    r8=rax+0xD2E9EE7E83C4285B;

    return _lrotl (r8, r8 % 60); // rotate left
};
```

:

```
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <intrin.h>

#define C1 0x5D7E0D1F2E0F1F84
#define C2 0x388D76AEE8CB1500
#define C3 0xD2E9EE7E83C4285B

uint64_t hash(uint64_t v)
{
    v*=C1;
    v=_lrotr(v, v&0xF); // rotate right
    v^=C2;
    v=_lrotl(v, v&0xF); // rotate left
```

```

        v+=C3;
        v=_lrotl(v, v % 60); // rotate left
        return v;
    }

int main()
{
    printf ("%llu\n", hash(...));
}

```

.

77.2

Microsoft Research Z3¹.

```

1 from z3 import *
2
3 C1=0x5D7E0D1F2E0F1F84
4 C2=0x388D76AEE8CB1500
5 C3=0xD2E9EE7E83C4285B
6
7 inp, i1, i2, i3, i4, i5, i6, outp = BitVecs('inp i1 i2 i3 i4 i5',
     ↴ i6 outp', 64)
8
9 s = Solver()
10 s.add(i1==inp*C1)
11 s.add(i2==RotateRight (i1, i1 & 0xF))
12 s.add(i3==i2 ^ C2)
13 s.add(i4==RotateLeft(i3, i3 & 0xF))
14 s.add(i5==i4 + C3)
15 s.add(outp==RotateLeft (i5, URem(i5, 60)))
16
17 s.add(outp==10816636949158156260)
18
19 print s.check()
20 m=s.model()
21 print m
22 print (" inp=0x%X" % m[inp].as_long())
23 print ("outp=0x%X" % m[outp].as_long())

```

¹<http://go.yurichev.com/17314>

.
..i1..i6 .

10..15. 10816636949158156260.

.

RotateRight, RotateLeft, URem .

:

```
...>python.exe 1.py
sat
[i1 = 3959740824832824396,
 i3 = 8957124831728646493,
 i5 = 10816636949158156260,
 inp = 1364123924608584563,
 outp = 10816636949158156260,
 i4 = 14065440378185297801,
 i2 = 4954926323707358301]
inp=0x12EE577B63E80B73
outp=0x961C69FF0AEFD7E4
```

«sat» «satisfiable», . . . 0x12EE577B63E80B73 .

. :

```
1 from z3 import *
2
3 C1=0x5D7E0D1F2E0F1F84
4 C2=0x388D76AEE8CB1500
5 C3=0xD2E9EE7E83C4285B
6
7 inp, i1, i2, i3, i4, i5, i6, outp = BitVecs('inp i1 i2 i3 i4 i5,
8     ↴ i6 outp', 64)
9
10 s = Solver()
11 s.add(i1==inp*C1)
12 s.add(i2==RotateRight (i1, i1 & 0xF))
13 s.add(i3==i2 ^ C2)
14 s.add(i4==RotateLeft(i3, i3 & 0xF))
15 s.add(i5==i4 + C3)
16 s.add(outp==RotateLeft (i5, URem(i5, 60)))
17 s.add(outp==10816636949158156260)
18 s.add(inp!=0x12EE577B63E80B73)
19
20 print s.check()
```

```

22 m=s.model()
23 print m
24 print (" inp=0x%X" % m[inp].as_long())
25 print ("outp=0x%X" % m[outp].as_long())

```

:

```

...>python.exe 2.py
sat
[i1 = 3959740824832824396,
 i3 = 8957124831728646493,
 i5 = 10816636949158156260,
 inp = 10587495961463360371,
 outp = 10816636949158156260,
 i4 = 14065440378185297801,
 i2 = 4954926323707358301]
inp=0x92EE577B63E80B73
outp=0x961C69FF0AEFD7E4

```

.

```

1 from z3 import *
2
3 C1=0x5D7E0D1F2E0F1F84
4 C2=0x388D76AEE8CB1500
5 C3=0xD2E9EE7E83C4285B
6
7 inp, i1, i2, i3, i4, i5, i6, outp = BitVecs('inp i1 i2 i3 i4 i5',
     ↴ i6 outp', 64)
8
9 s = Solver()
10 s.add(i1==inp*C1)
11 s.add(i2==RotateRight (i1, i1 & 0xF))
12 s.add(i3==i2 ^ C2)
13 s.add(i4==RotateLeft(i3, i3 & 0xF))
14 s.add(i5==i4 + C3)
15 s.add(outp==RotateLeft (i5, URem(i5, 60)))
16
17 s.add(outp==10816636949158156260)
18
19 # copypasted from http://stackoverflow.com/questions/11867611/
     ↴ z3py-checking-all-solutions-for-equation
20 result=[]
21 while True:
22     if s.check() == sat:
23         m = s.model()
24         print m[inp]

```

```

25     result.append(m)
26     # Create a new constraint the blocks the current model
27     block = []
28     for d in m:
29         # d is a declaration
30         if d.arity() > 0:
31             raise Z3Exception("uninterpreted functions are ↴
32             ↴ not supported")
33             # create a constant from declaration
34             c=d()
35             if is_array(c) or c.sort().kind() == ↴
36             ↴ Z3_UNINTERPRETED_SORT:
37                 raise Z3Exception("arrays and uninterpreted ↴
38                 ↴ sorts are not supported")
39                 block.append(c != m[d])
40                 s.add(Or(block))
41             else:
42                 print "results total=",len(result)
43                 break

```

:

```

1364123924608584563
1234567890
9223372038089343698
4611686019661955794
13835058056516731602
3096040143925676201
12319412180780452009
7707726162353064105
16931098199207839913
1906652839273745429
11130024876128521237
15741710894555909141
6518338857701133333
5975809943035972467
15199181979890748275
10587495961463360371
results total= 16

```

0x92EE577B63E80B73 .

1234567890.

. ?

outp :

```

1 from z3 import *
2
3 C1=0x5D7E0D1F2E0F1F84
4 C2=0x388D76AEE8CB1500
5 C3=0xD2E9EE7E83C4285B
6
7 inp, i1, i2, i3, i4, i5, i6, outp = BitVecs('inp i1 i2 i3 i4 i5,'
8     ↴ i6 outp', 64)
9
10 s = Solver()
11 s.add(i1==inp*C1)
12 s.add(i2==RotateRight (i1, i1 & 0xF))
13 s.add(i3==i2 ^ C2)
14 s.add(i4==RotateLeft(i3, i3 & 0xF))
15 s.add(i5==i4 + C3)
16 s.add(outp==RotateLeft (i5, URem(i5, 60)))
17 s.add(outp & 0xFFFFFFFF == inp & 0xFFFFFFFF)
18
19 print s.check()
20 m=s.model()
21 print m
22 print (" inp=0x%X" % m[inp].as_long())
23 print ("outp=0x%X" % m[outp].as_long())

```

:

```

sat
[i1 = 14869545517796235860,
 i3 = 8388171335828825253,
 i5 = 6918262285561543945,
 inp = 1370377541658871093,
 outp = 14543180351754208565,
 i4 = 10167065714588685486,
 i2 = 5541032613289652645]
inp=0x13048F1D12C00535
outp=0xC9D3C17A12C00535

```

0x1234:

```

1 from z3 import *
2
3 C1=0x5D7E0D1F2E0F1F84
4 C2=0x388D76AEE8CB1500
5 C3=0xD2E9EE7E83C4285B
6
7 inp, i1, i2, i3, i4, i5, i6, outp = BitVecs('inp i1 i2 i3 i4 i5,'
8     ↴ i6 outp', 64)

```

```

8 s = Solver()
9 s.add(i1==inp*C1)
10 s.add(i2==RotateRight (i1, i1 & 0xF))
11 s.add(i3==i2 ^ C2)
12 s.add(i4==RotateLeft(i3, i3 & 0xF))
13 s.add(i5==i4 + C3)
14 s.add(outp==RotateLeft (i5, URem(i5, 60)))
15
16
17 s.add(outp & 0xFFFFFFFF == inp & 0xFFFFFFFF)
18 s.add(outp & 0xFFFF == 0x1234)
19
20 print s.check()
21 m=s.model()
22 print m
23 print (" inp=0x%X" % m[inp].as_long())
24 print ("outp=0x%X" % m[outp].as_long())

```

:

```

sat
[i1 = 2834222860503985872,
 i3 = 2294680776671411152,
 i5 = 17492621421353821227,
 inp = 461881484695179828,
 outp = 419247225543463476,
 i4 = 2294680776671411152,
 i2 = 2834222860503985872]
inp=0x668EEC35F961234
outp=0x5D177215F961234

```

? AES, RSA,..

[Yur12].

Capítulo 78

: [Yur12].

78.1 #1: MacOS Classic y PowerPC

MacOS Classic ¹, PowerPC..

"Invalid Security Device". .

.

IDA "PEF (Mac OS or Be OS executable)" ().

:

```
...
seg000:000C87FC 38 60 00 01          li      %r3, 1
seg000:000C8800 48 03 93 41          bl      check1
seg000:000C8804 60 00 00 00          nop
seg000:000C8808 54 60 06 3F          clrlwi. %r0, %r3, ↵
                                     ↴ 24
seg000:000C880C 40 82 00 40          bne     OK
seg000:000C8810 80 62 9F D8          lwz      %r3, ↵
                                     ↴ TC_aInvalidSecurityDevice
...
```

...

check1().BL *Branch Link*..

[SK95].

CLRLWI. [IBM00]. MOVZX en x86 (15.1.1 on page 247), BNE².

check1():

```
seg000:00101B40           check1: # CODE XREF: seg000:00063✓
    ↳ E7Cp
seg000:00101B40           # sub_64070+160p ...
seg000:00101B40
seg000:00101B40           .set arg_8, 8
seg000:00101B40
seg000:00101B40 7C 08 02 A6      mflr    %r0
seg000:00101B44 90 01 00 08      stw     %r0, arg_8(%sp)
seg000:00101B48 94 21 FF C0      stwu   %sp, -0x40(%sp)
seg000:00101B4C 48 01 6B 39      bl      check2
seg000:00101B50 60 00 00 00      nop
seg000:00101B54 80 01 00 48      lwz     %r0, 0x40+arg_8(%sp)✓
    ↳ )
seg000:00101B58 38 21 00 40      addi   %sp, %sp, 0x40
seg000:00101B5C 7C 08 03 A6      mtlr   %r0
seg000:00101B60 4E 80 00 20      blr
seg000:00101B60           # End of function check1
```

.thunk function: check1() check2().

BLR³. link register, ARM.

check2():

```
seg000:00118684           check2: # CODE XREF: check1+Cp
seg000:00118684
seg000:00118684           .set var_18, -0x18
seg000:00118684           .set var_C, -0xC
seg000:00118684           .set var_8, -8
seg000:00118684           .set var_4, -4
seg000:00118684           .set arg_8, 8
seg000:00118684
seg000:00118684 93 E1 FF FC      stw     %r31, var_4(%sp)
seg000:00118688 7C 08 02 A6      mflr   %r0
seg000:0011868C 83 E2 95 A8      lwz     %r31, off_1485E8 # ✓
    ↳ dword_24B704
seg000:00118690           .using dword_24B704, %r31
seg000:00118690 93 C1 FF F8      stw     %r30, var_8(%sp)
seg000:00118694 93 A1 FF F4      stw     %r29, var_C(%sp)
seg000:00118698 7C 7D 1B 78      mr      %r29, %r3
seg000:0011869C 90 01 00 08      stw     %r0, arg_8(%sp)
seg000:001186A0 54 60 06 3E      clrlwi %r0, %r3, 24
```

²(PowerPC, ARM) Branch if Not Equal

³(PowerPC) Branch to Link Register

```

seg000:001186A4 28 00 00 01    cmplwi  %r0, 1
seg000:001186A8 94 21 FF B0    stwu    %sp, -0x50(%sp)
seg000:001186AC 40 82 00 0C    bne     loc_1186B8
seg000:001186B0 38 60 00 01    li      %r3, 1
seg000:001186B4 48 00 00 6C    b       exit
seg000:001186B8
seg000:001186B8          loc_1186B8: # CODE XREF: check2+28j
seg000:001186B8 48 00 03 D5    bl      sub_118A8C
seg000:001186BC 60 00 00 00    nop
seg000:001186C0 3B C0 00 00    li      %r30, 0
seg000:001186C4
seg000:001186C4          skip:    # CODE XREF: check2+94j
seg000:001186C4 57 C0 06 3F    clrlwi. %r0, %r30, 24
seg000:001186C8 41 82 00 18    beq     loc_1186E0
seg000:001186CC 38 61 00 38    addi    %r3, %sp, 0x50+var_18
seg000:001186D0 80 9F 00 00    lwz     %r4, dword_24B704
seg000:001186D4 48 00 C0 55    bl      .RBEFINDNEXT
seg000:001186D8 60 00 00 00    nop
seg000:001186DC 48 00 00 1C    b       loc_1186F8
seg000:001186E0
seg000:001186E0          loc_1186E0: # CODE XREF: check2+44j
seg000:001186E0 80 BF 00 00    lwz     %r5, dword_24B704
seg000:001186E4 38 81 00 38    addi   %r4, %sp, 0x50+var_18
seg000:001186E8 38 60 08 C2    li      %r3, 0x1234
seg000:001186EC 48 00 BF 99    bl      .RBEFINDFIRST
seg000:001186F0 60 00 00 00    nop
seg000:001186F4 3B C0 00 01    li      %r30, 1
seg000:001186F8
seg000:001186F8          loc_1186F8: # CODE XREF: check2+58j
seg000:001186F8 54 60 04 3F    clrlwi. %r0, %r3, 16
seg000:001186FC 41 82 00 0C    beq     must_jump
seg000:00118700 38 60 00 00    li      %r3, 0           # error
seg000:00118704 48 00 00 1C    b       exit
seg000:00118708
seg000:00118708          must_jump: # CODE XREF: check2+78j
seg000:00118708 7F A3 EB 78    mr      %r3, %r29
seg000:0011870C 48 00 00 31    bl      check3
seg000:00118710 60 00 00 00    nop
seg000:00118714 54 60 06 3F    clrlwi. %r0, %r3, 24
seg000:00118718 41 82 FF AC    beq     skip
seg000:0011871C 38 60 00 01    li      %r3, 1
seg000:00118720
seg000:00118720          exit:    # CODE XREF: check2+30j
seg000:00118720          # check2+80j
seg000:00118720 80 01 00 58    lwz     %r0, 0x50+arg_8(%sp)
seg000:00118724 38 21 00 50    addi   %sp, %sp, 0x50
seg000:00118728 83 E1 FF FC    lwz     %r31, var_4(%sp)

```

```

seg000:0011872C 7C 08 03 A6    mtlr    %r0
seg000:00118730 83 C1 FF F8    lwz     %r30, var_8(%sp)
seg000:00118734 83 A1 FF F4    lwz     %r29, var_C(%sp)
seg000:00118738 4E 80 00 20    blr
seg000:00118738                 # End of function check2

```

.GetNextDeviceViaUSB(), .USBSendPKT(),.

.GetNextEve3Device() Sentinel Eve3 .

·

li %r3, 1 li %r3, 0 (*Load Immediate*,). 0x001186B0.

:.RBEFINDFIRST() .RBEFINDNEXT() .

N.B.: clrlwi. %r0, %r3, 16, .RBEFINDFIRST() .

B (*branch*) .

BEQ BNE.

check3():

```

seg000:0011873C           check3: # CODE XREF: check2+88p
seg000:0011873C
seg000:0011873C           .set var_18, -0x18
seg000:0011873C           .set var_C, -0xC
seg000:0011873C           .set var_8, -8
seg000:0011873C           .set var_4, -4
seg000:0011873C           .set arg_8, 8
seg000:0011873C
seg000:0011873C 93 E1 FF FC   stw     %r31, var_4(%sp)
seg000:00118740 7C 08 02 A6   mflr    %r0
seg000:00118744 38 A0 00 00   li      %r5, 0
seg000:00118748 93 C1 FF F8   stw     %r30, var_8(%sp)
seg000:0011874C 83 C2 95 A8   lwz     %r30, off_1485E8 # ↴
                                ↓ dword_24B704
seg000:00118750           .using dword_24B704, %r30
seg000:00118750 93 A1 FF F4   stw     %r29, var_C(%sp)
seg000:00118754 3B A3 00 00   addi    %r29, %r3, 0
seg000:00118758 38 60 00 00   li      %r3, 0
seg000:0011875C 90 01 00 08   stw     %r0, arg_8(%sp)
seg000:00118760 94 21 FF B0   stwu   %sp, -0x50(%sp)
seg000:00118764 80 DE 00 00   lwz     %r6, dword_24B704
seg000:00118768 38 81 00 38   addi    %r4, %sp, 0x50+var_18
seg000:0011876C 48 00 C0 5D   bl      .RBEREAD
seg000:00118770 60 00 00 00   nop
seg000:00118774 54 60 04 3F   clrlwi. %r0, %r3, 16
seg000:00118778 41 82 00 0C   beq    loc_118784

```

seg000:0011877C	38	60	00	00	li	%r3,	0	
seg000:00118780	48	00	02	F0	b	exit		
seg000:00118784								
seg000:00118784					loc_118784:	# CODE XREF: check3+3Cj		
seg000:00118784	A0	01	00	38	lhz	%r0,	0x50+var_18(%sp)	
seg000:00118788	28	00	04	B2	cmpwi	%r0,	0x1100	
seg000:0011878C	41	82	00	0C	beq	loc_118798		
seg000:00118790	38	60	00	00	li	%r3,	0	
seg000:00118794	48	00	02	DC	b	exit		
seg000:00118798								
seg000:00118798					loc_118798:	# CODE XREF: check3+50j		
seg000:00118798	80	DE	00	00	lwz	%r6,	dword_24B704	
seg000:0011879C	38	81	00	38	addi	%r4,	%sp, 0x50+var_18	
seg000:001187A0	38	60	00	01	li	%r3,	1	
seg000:001187A4	38	A0	00	00	li	%r5,	0	
seg000:001187A8	48	00	C0	21	bl	.RBEREAD		
seg000:001187AC	60	00	00	00	nop			
seg000:001187B0	54	60	04	3F	clrlwi.	%r0,	%r3, 16	
seg000:001187B4	41	82	00	0C	beq	loc_1187C0		
seg000:001187B8	38	60	00	00	li	%r3,	0	
seg000:001187BC	48	00	02	B4	b	exit		
seg000:001187C0								
seg000:001187C0					loc_1187C0:	# CODE XREF: check3+78j		
seg000:001187C0	A0	01	00	38	lhz	%r0,	0x50+var_18(%sp)	
seg000:001187C4	28	00	06	4B	cmpwi	%r0,	0x09AB	
seg000:001187C8	41	82	00	0C	beq	loc_1187D4		
seg000:001187CC	38	60	00	00	li	%r3,	0	
seg000:001187D0	48	00	02	A0	b	exit		
seg000:001187D4								
seg000:001187D4					loc_1187D4:	# CODE XREF: check3+8Cj		
seg000:001187D4	4B	F9	F3	D9	bl	sub_B7BAC		
seg000:001187D8	60	00	00	00	nop			
seg000:001187DC	54	60	06	3E	clrlwi	%r0,	%r3, 24	
seg000:001187E0	2C	00	00	05	cmpwi	%r0,	5	
seg000:001187E4	41	82	01	00	beq	loc_1188E4		
seg000:001187E8	40	80	00	10	bge	loc_1187F8		
seg000:001187EC	2C	00	00	04	cmpwi	%r0,	4	
seg000:001187F0	40	80	00	58	bge	loc_118848		
seg000:001187F4	48	00	01	8C	b	loc_118980		
seg000:001187F8								
seg000:001187F8					loc_1187F8:	# CODE XREF: check3+ACj		
seg000:001187F8	2C	00	00	0B	cmpwi	%r0,	0xB	
seg000:001187FC	41	82	00	08	beq	loc_118804		
seg000:00118800	48	00	01	80	b	loc_118980		
seg000:00118804								
seg000:00118804					loc_118804:	# CODE XREF: check3+C0j		
seg000:00118804	80	DE	00	00	lwz	%r6,	dword_24B704	

```

seg000:00118808 38 81 00 38    addi    %r4, %sp, 0x50+var_18
seg000:0011880C 38 60 00 08    li      %r3, 8
seg000:00118810 38 A0 00 00    li      %r5, 0
seg000:00118814 48 00 BF B5    bl      .RBREAD
seg000:00118818 60 00 00 00    nop
seg000:0011881C 54 60 04 3F    clrlwi. %r0, %r3, 16
seg000:00118820 41 82 00 0C    beq    loc_11882C
seg000:00118824 38 60 00 00    li      %r3, 0
seg000:00118828 48 00 02 48    b       exit
seg000:0011882C
seg000:0011882C          loc_11882C: # CODE XREF: check3+E4j
seg000:0011882C A0 01 00 38    lhz    %r0, 0x50+var_18(%sp)
seg000:00118830 28 00 11 30    cmplwi %r0, 0xFEAO
seg000:00118834 41 82 00 0C    beq    loc_118840
seg000:00118838 38 60 00 00    li      %r3, 0
seg000:0011883C 48 00 02 34    b       exit
seg000:00118840
seg000:00118840          loc_118840: # CODE XREF: check3+F8j
seg000:00118840 38 60 00 01    li      %r3, 1
seg000:00118844 48 00 02 2C    b       exit
seg000:00118848
seg000:00118848          loc_118848: # CODE XREF: check3+B4j
seg000:00118848 80 DE 00 00    lwz     %r6, dword_24B704
seg000:0011884C 38 81 00 38    addi   %r4, %sp, 0x50+var_18
seg000:00118850 38 60 00 0A    li      %r3, 0xA
seg000:00118854 38 A0 00 00    li      %r5, 0
seg000:00118858 48 00 BF 71    bl      .RBREAD
seg000:0011885C 60 00 00 00    nop
seg000:00118860 54 60 04 3F    clrlwi. %r0, %r3, 16
seg000:00118864 41 82 00 0C    beq    loc_118870
seg000:00118868 38 60 00 00    li      %r3, 0
seg000:0011886C 48 00 02 04    b       exit
seg000:00118870
seg000:00118870          loc_118870: # CODE XREF: check3+128j
    ↳ j
seg000:00118870 A0 01 00 38    lhz    %r0, 0x50+var_18(%sp)
seg000:00118874 28 00 03 F3    cmplwi %r0, 0xA6E1
seg000:00118878 41 82 00 0C    beq    loc_118884
seg000:0011887C 38 60 00 00    li      %r3, 0
seg000:00118880 48 00 01 F0    b       exit
seg000:00118884
seg000:00118884          loc_118884: # CODE XREF: check3+13j
    ↳ Cj
seg000:00118884 57 BF 06 3E    clrlwi %r31, %r29, 24
seg000:00118888 28 1F 00 02    cmplwi %r31, 2
seg000:0011888C 40 82 00 0C    bne    loc_118898
seg000:00118890 38 60 00 01    li      %r3, 1

```

```

seg000:00118894 48 00 01 DC    b      exit
seg000:00118898
seg000:00118898          loc_118898: # CODE XREF: check3+150✓
    ↳ j
seg000:00118898 80 DE 00 00    lwz      %r6, dword_24B704
seg000:0011889C 38 81 00 38    addi     %r4, %sp, 0x50+var_18
seg000:001188A0 38 60 00 0B    li       %r3, 0xB
seg000:001188A4 38 A0 00 00    li       %r5, 0
seg000:001188A8 48 00 BF 21    bl       .RBREAD
seg000:001188AC 60 00 00 00    nop
seg000:001188B0 54 60 04 3F    clrlwi  %r0, %r3, 16
seg000:001188B4 41 82 00 0C    beq     loc_1188C0
seg000:001188B8 38 60 00 00    li       %r3, 0
seg000:001188BC 48 00 01 B4    b       exit
seg000:001188C0
seg000:001188C0          loc_1188C0: # CODE XREF: check3+178✓
    ↳ j
seg000:001188C0 A0 01 00 38    lhz      %r0, 0x50+var_18(%sp)
seg000:001188C4 28 00 23 1C    cmplwi  %r0, 0x1C20
seg000:001188C8 41 82 00 0C    beq     loc_1188D4
seg000:001188CC 38 60 00 00    li       %r3, 0
seg000:001188D0 48 00 01 A0    b       exit
seg000:001188D4
seg000:001188D4          loc_1188D4: # CODE XREF: check3+18✓
    ↳ Cj
seg000:001188D4 28 1F 00 03    cmplwi  %r31, 3
seg000:001188D8 40 82 01 94    bne     error
seg000:001188DC 38 60 00 01    li       %r3, 1
seg000:001188E0 48 00 01 90    b       exit
seg000:001188E4
seg000:001188E4          loc_1188E4: # CODE XREF: check3+A8j
seg000:001188E4 80 DE 00 00    lwz      %r6, dword_24B704
seg000:001188E8 38 81 00 38    addi     %r4, %sp, 0x50+var_18
seg000:001188EC 38 60 00 0C    li       %r3, 0xC
seg000:001188F0 38 A0 00 00    li       %r5, 0
seg000:001188F4 48 00 BE D5    bl       .RBREAD
seg000:001188F8 60 00 00 00    nop
seg000:001188FC 54 60 04 3F    clrlwi  %r0, %r3, 16
seg000:00118900 41 82 00 0C    beq     loc_11890C
seg000:00118904 38 60 00 00    li       %r3, 0
seg000:00118908 48 00 01 68    b       exit
seg000:0011890C
seg000:0011890C          loc_11890C: # CODE XREF: check3+1✓
    ↳ C4j
seg000:0011890C A0 01 00 38    lhz      %r0, 0x50+var_18(%sp)
seg000:00118910 28 00 1F 40    cmplwi  %r0, 0x40FF
seg000:00118914 41 82 00 0C    beq     loc_118920

```

```

seg000:00118918 38 60 00 00    li      %r3, 0
seg000:0011891C 48 00 01 54    b       exit
seg000:00118920
seg000:00118920          loc_118920: # CODE XREF: check3+1 ↴
    ↳ D8j
seg000:00118920 57 BF 06 3E    clrlwi  %r31, %r29, 24
seg000:00118924 28 1F 00 02    cmplwi  %r31, 2
seg000:00118928 40 82 00 0C    bne     loc_118934
seg000:0011892C 38 60 00 01    li      %r3, 1
seg000:00118930 48 00 01 40    b       exit
seg000:00118934
seg000:00118934          loc_118934: # CODE XREF: check3+1 ↴
    ↳ ECj
seg000:00118934 80 DE 00 00    lwz     %r6, dword_24B704
seg000:00118938 38 81 00 38    addi    %r4, %sp, 0x50+var_18
seg000:0011893C 38 60 00 0D    li      %r3, 0xD
seg000:00118940 38 A0 00 00    li      %r5, 0
seg000:00118944 48 00 BE 85    bl     .RBREAD
seg000:00118948 60 00 00 00    nop
seg000:0011894C 54 60 04 3F    clrlwi. %r0, %r3, 16
seg000:00118950 41 82 00 0C    beq    loc_11895C
seg000:00118954 38 60 00 00    li      %r3, 0
seg000:00118958 48 00 01 18    b       exit
seg000:0011895C
seg000:0011895C          loc_11895C: # CODE XREF: check3+214 ↴
    ↳ j
seg000:0011895C A0 01 00 38    lhz     %r0, 0x50+var_18(%sp)
seg000:00118960 28 00 07 CF    cmplwi %r0, 0xFC7
seg000:00118964 41 82 00 0C    beq    loc_118970
seg000:00118968 38 60 00 00    li      %r3, 0
seg000:0011896C 48 00 01 04    b       exit
seg000:00118970
seg000:00118970          loc_118970: # CODE XREF: check3+228 ↴
    ↳ j
seg000:00118970 28 1F 00 03    cmplwi %r31, 3
seg000:00118974 40 82 00 F8    bne     error
seg000:00118978 38 60 00 01    li      %r3, 1
seg000:0011897C 48 00 00 F4    b       exit
seg000:00118980
seg000:00118980          loc_118980: # CODE XREF: check3+B8j
seg000:00118980          # check3+C4j
seg000:00118980 80 DE 00 00    lwz     %r6, dword_24B704
seg000:00118984 38 81 00 38    addi   %r4, %sp, 0x50+var_18
seg000:00118988 3B E0 00 00    li      %r31, 0
seg000:0011898C 38 60 00 04    li      %r3, 4
seg000:00118990 38 A0 00 00    li      %r5, 0
seg000:00118994 48 00 BE 35    bl     .RBREAD

```

```

seg000:00118998 60 00 00 00    nop
seg000:0011899C 54 60 04 3F    clrlwi. %r0, %r3, 16
seg000:001189A0 41 82 00 0C    beq     loc_1189AC
seg000:001189A4 38 60 00 00    li      %r3, 0
seg000:001189A8 48 00 00 C8    b       exit
seg000:001189AC
seg000:001189AC           loc_1189AC: # CODE XREF: check3+264 ↴
    ↳ j
seg000:001189AC A0 01 00 38    lhz     %r0, 0x50+var_18(%sp)
seg000:001189B0 28 00 1D 6A    cmplwi %r0, 0xAED0
seg000:001189B4 40 82 00 0C    bne     loc_1189C0
seg000:001189B8 3B E0 00 01    li      %r31, 1
seg000:001189BC 48 00 00 14    b       loc_1189D0
seg000:001189C0
seg000:001189C0           loc_1189C0: # CODE XREF: check3+278 ↴
    ↳ j
seg000:001189C0 28 00 18 28    cmplwi %r0, 0x2818
seg000:001189C4 41 82 00 0C    beq     loc_1189D0
seg000:001189C8 38 60 00 00    li      %r3, 0
seg000:001189CC 48 00 00 A4    b       exit
seg000:001189D0
seg000:001189D0           loc_1189D0: # CODE XREF: check3+280 ↴
    ↳ j
seg000:001189D0           # check3+288j
seg000:001189D0 57 A0 06 3E    clrlwi %r0, %r29, 24
seg000:001189D4 28 00 00 02    cmplwi %r0, 2
seg000:001189D8 40 82 00 20    bne     loc_1189F8
seg000:001189DC 57 E0 06 3F    clrlwi. %r0, %r31, 24
seg000:001189E0 41 82 00 10    beq     good2
seg000:001189E4 48 00 4C 69    bl      sub_11D64C
seg000:001189E8 60 00 00 00    nop
seg000:001189EC 48 00 00 84    b       exit
seg000:001189F0
seg000:001189F0           good2:   # CODE XREF: check3+2 ↴
    ↳ A4j
seg000:001189F0 38 60 00 01    li      %r3, 1
seg000:001189F4 48 00 00 7C    b       exit
seg000:001189F8
seg000:001189F8           loc_1189F8: # CODE XREF: check3+29 ↴
    ↳ Cj
seg000:001189F8 80 DE 00 00    lwz     %r6, dword_24B704
seg000:001189FC 38 81 00 38    addi   %r4, %sp, 0x50+var_18
seg000:00118A00 38 60 00 05    li      %r3, 5
seg000:00118A04 38 A0 00 00    li      %r5, 0
seg000:00118A08 48 00 BD C1    bl      .RBREAD
seg000:00118A0C 60 00 00 00    nop
seg000:00118A10 54 60 04 3F    clrlwi. %r0, %r3, 16

```

```

seg000:00118A14 41 82 00 0C    beq      loc_118A20
seg000:00118A18 38 60 00 00    li       %r3, 0
seg000:00118A1C 48 00 00 54    b        exit
seg000:00118A20
seg000:00118A20          loc_118A20: # CODE XREF: check3+2
    ↳ D8j
seg000:00118A20 A0 01 00 38    lhz      %r0, 0x50+var_18(%sp)
seg000:00118A24 28 00 11 D3    cmplwi   %r0, 0xD300
seg000:00118A28 40 82 00 0C    bne     loc_118A34
seg000:00118A2C 3B E0 00 01    li       %r31, 1
seg000:00118A30 48 00 00 14    b        good1
seg000:00118A34
seg000:00118A34          loc_118A34: # CODE XREF: check3+2
    ↳ ECj
seg000:00118A34 28 00 1A EB    cmplwi   %r0, 0xEBA1
seg000:00118A38 41 82 00 0C    beq     good1
seg000:00118A3C 38 60 00 00    li       %r3, 0
seg000:00118A40 48 00 00 30    b        exit
seg000:00118A44
seg000:00118A44          good1:   # CODE XREF: check3+2
    ↳ F4j
seg000:00118A44
seg000:00118A44 57 A0 06 3E    clrlwi   %r0, %r29, 24
seg000:00118A48 28 00 00 03    cmplwi   %r0, 3
seg000:00118A4C 40 82 00 20    bne     error
seg000:00118A50 57 E0 06 3F    clrlwi. %r0, %r31, 24
seg000:00118A54 41 82 00 10    beq     good
seg000:00118A58 48 00 4B F5    bl      sub_11D64C
seg000:00118A5C 60 00 00 00    nop
seg000:00118A60 48 00 00 10    b        exit
seg000:00118A64
seg000:00118A64          good:   # CODE XREF: check3+318
    ↳ j
seg000:00118A64 38 60 00 01    li       %r3, 1
seg000:00118A68 48 00 00 08    b        exit
seg000:00118A6C
seg000:00118A6C          error:   # CODE XREF: check3+19
    ↳ Cj
seg000:00118A6C
seg000:00118A6C 38 60 00 00    li       # check3+238j ...
seg000:00118A70
seg000:00118A70          exit:    # CODE XREF: check3+44j
seg000:00118A70
seg000:00118A70 80 01 00 58    lwz      %r0, 0x50+arg_8(%sp)
seg000:00118A74 38 21 00 50    addi    %sp, %sp, 0x50
seg000:00118A78 83 E1 FF FC    lwz      %r31, var_4(%sp)
seg000:00118A7C 7C 08 03 A6    mtlr    %r0

```

```
seg000:00118A80 83 C1 FF F8    lwz      %r30, var_8(%sp)
seg000:00118A84 83 A1 FF F4    lwz      %r29, var_C(%sp)
seg000:00118A88 4E 80 00 20   blr
seg000:00118A88                 # End of function check3
```

.RBEREAD(). CMPLWI.

.RBEREAD(): 0, 1, 8, 0xA, 0xB, 0xC, 0xD, 4, 5. ?

Sentinel Eve3!

.RBEREAD(),.

.

0x001186FC 0x48 y 0 BEQ B (): [[IBM00](#)].

0x00118718 0x60 y NOP:..

.

.

78.2 #2: SCO OpenServer

SCO OpenServer 1997 .

: «Copyright 1989, Rainbow Technologies, Inc., Irvine, CA» y «Sentinel Integrated Driver Ver. 3.0 ».

:

```
/dev/rbs18
/dev/rbs19
/dev/rbs110
```

SCO OpenServer.

:

```
.text:00022AB8      public SSQC
.text:00022AB8 SSQC    proc near ; CODE XREF: SSQ+7p
.text:00022AB8
.text:00022AB8 var_44 = byte ptr -44h
.text:00022AB8 var_29 = byte ptr -29h
.text:00022AB8 arg_0  = dword ptr  8
.text:00022AB8         push    ebp
```

```

.text:00022AB9      mov    ebp, esp
.text:00022ABB      sub    esp, 44h
.text:00022ABE      push   edi
.text:00022ABF      mov    edi, offset unk_4035D0
.text:00022AC4      push   esi
.text:00022AC5      mov    esi, [ebp+arg_0]
.text:00022AC8      push   ebx
.text:00022AC9      push   esi
.text:00022ACA      call   strlen
.text:00022ACF      add    esp, 4
.text:00022AD2      cmp    eax, 2
.text:00022AD7      jnz   loc_22BA4
.text:00022ADD      inc    esi
.text:00022ADE      mov    al, [esi-1]
.text:00022AE1      movsx eax, al
.text:00022AE4      cmp    eax, '3'
.text:00022AE9      jz    loc_22B84
.text:00022AEF      cmp    eax, '4'
.text:00022AF4      jz    loc_22B94
.text:00022AFA      cmp    eax, '5'
.text:00022AFF      jnz   short loc_22B6B
.text:00022B01      movsx ebx, byte ptr [esi]
.text:00022B04      sub    ebx, '0'
.text:00022B07      mov    eax, 7
.text:00022B0C      add    eax, ebx
.text:00022B0E      push   eax
.text:00022B0F      lea    eax, [ebp+var_44]
.text:00022B12      push   offset aDevSlD ; "/dev/sl%d"
.text:00022B17      push   eax
.text:00022B18      call   nl_sprintf
.text:00022B1D      push   0           ; int
.text:00022B1F      push   offset aDevRbsl8 ; char *
.text:00022B24      call   _access
.text:00022B29      add    esp, 14h
.text:00022B2C      cmp    eax, 0FFFFFFFh
.text:00022B31      jz    short loc_22B48
.text:00022B33      lea    eax, [ebx+7]
.text:00022B36      push   eax
.text:00022B37      lea    eax, [ebp+var_44]
.text:00022B3A      push   offset aDevRbslD ; "/dev/rbsl%d"
.text:00022B3F      push   eax
.text:00022B40      call   nl_sprintf
.text:00022B45      add    esp, 0Ch
.text:00022B48      loc_22B48: ; CODE XREF: SSQC+79j
.text:00022B48      mov    edx, [edi]
.text:00022B4A      test  edx, edx

```

```
.text:00022B4C      jle     short loc_22B57
.text:00022B4E      push    edx          ; int
.text:00022B4F      call    _close
.text:00022B54      add    esp, 4
.text:00022B57
.text:00022B57 loc_22B57: ; CODE XREF: SSQC+94j
.text:00022B57      push    2             ; int
.text:00022B59      lea     eax, [ebp+var_44]
.text:00022B5C      push    eax          ; char *
.text:00022B5D      call    _open
.text:00022B62      add    esp, 8
.text:00022B65      test   eax, eax
.text:00022B67      mov    [edi], eax
.text:00022B69      jge     short loc_22B78
.text:00022B6B
.text:00022B6B loc_22B6B: ; CODE XREF: SSQC+47j
.text:00022B6B      mov    eax, 0FFFFFFFh
.text:00022B70      pop    ebx
.text:00022B71      pop    esi
.text:00022B72      pop    edi
.text:00022B73      mov    esp, ebp
.text:00022B75      pop    ebp
.text:00022B76      retn
.text:00022B78
.text:00022B78 loc_22B78: ; CODE XREF: SSQC+B1j
.text:00022B78      pop    ebx
.text:00022B79      pop    esi
.text:00022B7A      pop    edi
.text:00022B7B      xor    eax, eax
.text:00022B7D      mov    esp, ebp
.text:00022B7F      pop    ebp
.text:00022B80      retn
.text:00022B84
.text:00022B84 loc_22B84: ; CODE XREF: SSQC+31j
.text:00022B84      mov    al, [esi]
.text:00022B86      pop    ebx
.text:00022B87      pop    esi
.text:00022B88      pop    edi
.text:00022B89      mov    ds:byte_407224, al
.text:00022B8E      mov    esp, ebp
.text:00022B90      xor    eax, eax
.text:00022B92      pop    ebp
.text:00022B93      retn
.text:00022B94
.text:00022B94 loc_22B94: ; CODE XREF: SSQC+3Cj
.text:00022B94      mov    al, [esi]
.text:00022B96      pop    ebx
```

```

.text:00022B97      pop    esi
.text:00022B98      pop    edi
.text:00022B99      mov     ds:byte_407225, al
.text:00022B9E      mov     esp, ebp
.text:00022BA0      xor    eax, eax
.text:00022BA2      pop    ebp
.text:00022BA3      retn
.text:00022BA4
.loc_22BA4: ; CODE XREF: SSQC+1Fj
.text:00022BA4      movsx  eax, ds:byte_407225
.text:00022BAB      push   esi
.text:00022BAC      push   eax
.text:00022BAD      movsx  eax, ds:byte_407224
.text:00022BB4      push   eax
.text:00022BB5      lea    eax, [ebp+var_44]
.text:00022BB8      push   offset a46CCS    ; "46%c%c%s"
.text:00022BBD      push   eax
.text:00022BBE      call   nl_sprintf
.text:00022BC3      lea    eax, [ebp+var_44]
.text:00022BC6      push   eax
.text:00022BC7      call   strlen
.text:00022BCC      add    esp, 18h
.text:00022BCF      cmp    eax, 1Bh
.text:00022BD4      jle    short loc_22BDA
.text:00022BD6      mov    [ebp+var_29], 0
.text:00022BDA
.loc_22BDA: ; CODE XREF: SSQC+11Cj
.text:00022BDA      lea    eax, [ebp+var_44]
.text:00022BDD      push   eax
.text:00022BDE      call   strlen
.text:00022BE3      push   eax          ; unsigned int
.text:00022BE4      lea    eax, [ebp+var_44]
.text:00022BE7      push   eax          ; void *
.text:00022BE8      mov    eax, [edi]
.text:00022BEA      push   eax          ; int
.text:00022BEB      call   _write
.text:00022BF0      add    esp, 10h
.text:00022BF3      pop    ebx
.text:00022BF4      pop    esi
.text:00022BF5      pop    edi
.text:00022BF6      mov    esp, ebp
.text:00022BF8      pop    ebp
.text:00022BF9      retn
.text:00022BFA      db    0Eh dup(90h)
.text:00022BFA  SSQC  endp

```

SSQC() thunk function:

```
.text:0000DBE8    public SSQ
.text:0000DBE8 SSQ    proc near ; CODE XREF: sys_info+A9p
.text:0000DBE8          ; sys_info+CBp ...
.text:0000DBE8
.text:0000DBE8 arg_0  = dword ptr  8
.text:0000DBE8
.text:0000DBE8     push    ebp
.text:0000DBE9     mov     ebp, esp
.text:0000DBEB     mov     edx, [ebp+arg_0]
.text:0000DBEE     push    edx
.text:0000DBEF     call    SSQC
.text:0000DBF4     add     esp, 4
.text:0000DBF7     mov     esp, ebp
.text:0000DBF9     pop    ebp
.text:0000DBFA     retn
.text:0000DBFB SSQ    endp
```

SSQ() .

:

```
.data:0040169C _51_52_53      dd offset aPressAnyKeyT_0 ; DATA ↴
    ↳ XREF: init_sys+392r
.data:0040169C                  ; ↴
    ↳ sys_info+A1r
.data:0040169C                  ; "PRESS" ↴
    ↳ ANY KEY TO CONTINUE: "
.data:004016A0                  dd offset a51           ; "51"
.data:004016A4                  dd offset a52           ; "52"
.data:004016A8                  dd offset a53           ; "53"

...
.data:004016B8 _3C_or_3E      dd offset a3c           ; DATA ↴
    ↳ XREF: sys_info:loc_D67Br
.data:004016B8                  ; "3C"
.data:004016BC                  dd offset a3e           ; "3E"

; these names we gave to the labels:
.data:004016C0 answers1        dd 6B05h           ; DATA ↴
    ↳ XREF: sys_info+E7r
.data:004016C4                  dd 3D87h
.data:004016C8 answers2        dd 3Ch            ; DATA ↴
    ↳ XREF: sys_info+F2r
.data:004016CC                  dd 832h
```

```

.data:004016D0 _C_and_B        db 0Ch           ; DATA  ↴
    ↳ XREF: sys_info+BAr
.data:004016D0                 db               ; ↴
    ↳ sys_info:OKr
.data:004016D1 byte_4016D1     db 0Bh          ; DATA  ↴
    ↳ XREF: sys_info+FDr
.data:004016D2                 db 0             ; ↴
...
.text:0000D652                 xor  eax, eax
.text:0000D654                 mov   al, ds:ctl_port
.text:0000D659                 mov   ecx, _51_52_53[eax*4]
.text:0000D660                 push  ecx
.text:0000D661                 call  SSQ
.text:0000D666                 add   esp, 4
.text:0000D669                 cmp   eax, 0FFFFFFFh
.text:0000D66E                 jz    short loc_D6D1
.text:0000D670                 xor   ebx, ebx
.text:0000D672                 mov   al, _C_and_B
.text:0000D677                 test  al, al
.text:0000D679                 jz    short loc_D6C0
.text:0000D67B
.text:0000D67B loc_D67B: ; CODE XREF: sys_info+106j
.text:0000D67B                 mov   eax, _3C_or_3E[ebx*4]
.text:0000D682                 push  eax
.text:0000D683                 call  SSQ
.text:0000D688                 push  offset a4g      ; "4G"
.text:0000D68D                 call  SSQ
.text:0000D692                 push  offset a0123456789 ; ↴
    ↳ "0123456789"
.text:0000D697                 call  SSQ
.text:0000D69C                 add   esp, 0Ch
.text:0000D69F                 mov   edx, answers1[ebx*4]
.text:0000D6A6                 cmp   eax, edx
.text:0000D6A8                 jz    short OK
.text:0000D6AA                 mov   ecx, answers2[ebx*4]
.text:0000D6B1                 cmp   eax, ecx
.text:0000D6B3                 jz    short OK
.text:0000D6B5                 mov   al, byte_4016D1[ebx]
.text:0000D6BB                 inc   ebx
.text:0000D6BC                 test  al, al
.text:0000D6BE                 jnz   short loc_D67B
.text:0000D6C0
.text:0000D6C0 loc_D6C0: ; CODE XREF: sys_info+C1j
.text:0000D6C0                 inc   ds:ctl_port
.text:0000D6C6                 xor   eax, eax

```

```

.text:0000D6C8          mov     al, ds:ctl_port
.text:0000D6CD          cmp     eax, edi
.text:0000D6CF          jle     short loc_D652
.text:0000D6D1
.text:0000D6D1 loc_D6D1: ; CODE XREF: sys_info+98j
.text:0000D6D1           ; sys_info+B6j
.text:0000D6D1           mov     edx, [ebp+var_8]
.text:0000D6D4           inc     edx
.text:0000D6D5           mov     [ebp+var_8], edx
.text:0000D6D8           cmp     edx, 3
.text:0000D6DB           jle     loc_D641
.text:0000D6E1
.text:0000D6E1 loc_D6E1: ; CODE XREF: sys_info+16j
.text:0000D6E1           ; sys_info+51j ...
.text:0000D6E1           pop    ebx
.text:0000D6E2           pop    edi
.text:0000D6E3           mov    esp, ebp
.text:0000D6E5           pop    ebp
.text:0000D6E6           retn
.text:0000D6E8 OK:      ; CODE XREF: sys_info+F0j
.text:0000D6E8           ; sys_info+FBj
.text:0000D6E8           mov    al, _C_and_B[ebx]
.text:0000D6EE           pop    ebx
.text:0000D6EF           pop    edi
.text:0000D6F0           mov    ds:ctl_model, al
.text:0000D6F5           mov    esp, ebp
.text:0000D6F7           pop    ebp
.text:0000D6F8           retn
.text:0000D6F8 sys_info  endp

```

«3C» y «3E» Sentinel Pro .

: 34 on page 590.

. . . (SSQC()) . . .

.

51/52/53 . 3x/4x «family» .

"0123456789". .., 0xC o 0xB ctl_model.

"PRESS ANY KEY TO CONTINUE:", . ⁴.

ctl_mode.

:

```
.text:0000D708 prep_sys proc near ; CODE XREF: init_sys+46Ap
.text:0000D708
.text:0000D708 var_14      = dword ptr -14h
.text:0000D708 var_10      = byte ptr -10h
.text:0000D708 var_8       = dword ptr -8
.text:0000D708 var_2       = word ptr -2
.text:0000D708
.text:0000D708     push    ebp
.text:0000D709     mov     eax, ds:net_env
.text:0000D70E     mov     ebp, esp
.text:0000D710     sub     esp, 1Ch
.text:0000D713     test    eax, eax
.text:0000D715     jnz    short loc_D734
.text:0000D717     mov     al, ds:ctl_model
.text:0000D71C     test    al, al
.text:0000D71E     jnz    short loc_D77E
.text:0000D720     mov     [ebp+var_8], offset aIeCvulnvV\\\bOKG]T_
.text:0000D727     mov     edx, 7
.text:0000D72C     jmp    loc_D7E7

...
```

```
.text:0000D7E7 loc_D7E7: ; CODE XREF: prep_sys+24j
.text:0000D7E7          ; prep_sys+33j
.text:0000D7E7     push    edx
.text:0000D7E8     mov     edx, [ebp+var_8]
.text:0000D7EB     push    20h
.text:0000D7ED     push    edx
.text:0000D7EE     push    16h
.text:0000D7F0     call    err_warn
.text:0000D7F5     push    offset station_sem
.text:0000D7FA     call    ClosSem
.text:0000D7FF     call    startup_err
```

```
.
```

```
:
```

```
.text:0000A43C err_warn         proc near           ; CODE 
    ↳ XREF: prep_sys+E8p
.text:0000A43C
    ↳ prep_sys2+2Fp ...
.text:0000A43C
.text:0000A43C var_55          = byte ptr -55h
.text:0000A43C var_54          = byte ptr -54h
.text:0000A43C arg_0           = dword ptr 8
```

```

.text:0000A43C arg_4          = dword ptr 0Ch
.text:0000A43C arg_8          = dword ptr 10h
.text:0000A43C arg_C          = dword ptr 14h
.text:0000A43C
.text:0000A43C                push    ebp
.text:0000A43D                mov     ebp, esp
.text:0000A43F                sub     esp, 54h
.text:0000A442                push    edi
.text:0000A443                mov     ecx, [ebp+arg_8]
.text:0000A446                xor     edi, edi
.text:0000A448                test    ecx, ecx
.text:0000A44A                push    esi
.text:0000A44B                jle    short loc_A466
.text:0000A44D                mov     esi, [ebp+arg_C] ; key
.text:0000A450                mov     edx, [ebp+arg_4] ; ↵
    ↓ string
.text:0000A453
.text:0000A453 loc_A453:           ; CODE ↵
    ↓ XREF: err_warn+28j
.text:0000A453                xor     eax, eax
.text:0000A455                mov     al, [edx+edi]
.text:0000A458                xor     eax, esi
.text:0000A45A                add     esi, 3
.text:0000A45D                inc     edi
.text:0000A45E                cmp     edi, ecx
.text:0000A460                mov     [ebp+edi+var_55], al
.text:0000A464                jl    short loc_A453
.text:0000A466
.text:0000A466 loc_A466:           ; CODE ↵
    ↓ XREF: err_warn+Fj
.text:0000A466                mov     [ebp+edi+var_54], 0
.text:0000A46B                mov     eax, [ebp+arg_0]
.text:0000A46E                cmp     eax, 18h
.text:0000A473                jnz    short loc_A49C
.text:0000A475                lea     eax, [ebp+var_54]
.text:0000A478                push   eax
.text:0000A479                call   status_line
.text:0000A47E                add    esp, 4
.text:0000A481
.text:0000A481 loc_A481:           ; CODE ↵
    ↓ XREF: err_warn+72j
.text:0000A481                push   50h
.text:0000A483                push   0
.text:0000A485                lea    eax, [ebp+var_54]
.text:0000A488                push   eax
.text:0000A489                call   memset
.text:0000A48E                call   pcv_refresh

```

```

.text:0000A493          add    esp, 0Ch
.text:0000A496          pop    esi
.text:0000A497          pop    edi
.text:0000A498          mov    esp, ebp
.text:0000A49A          pop    ebp
.text:0000A49B          retn
.text:0000A49C          ; CODE 
    ↳ XREF: err_warn+37j
.text:0000A49C loc_A49C:
.text:0000A49C          push   0
.text:0000A49E          lea    eax, [ebp+var_54]
.text:0000A4A1          mov    edx, [ebp+arg_0]
.text:0000A4A4          push   edx
.text:0000A4A5          push   eax
.text:0000A4A6          call   pcv_lputs
.text:0000A4AB          add    esp, 0Ch
.text:0000A4AE          jmp    short loc_A481
.text:0000A4AE err_warn  endp

```

«offln» 0xFE81 y 0x12A9. timer().

```

.text:0000DA55 loc_DA55:                                ; CODE 
    ↳ XREF: sync_sys+24Cj
.text:0000DA55          push   offset aOffln     ; "offln"
    ↳ "
.text:0000DA5A          call   SSQ
.text:0000DA5F          add    esp, 4
.text:0000DA62          mov    dl, [ebx]
.text:0000DA64          mov    esi, eax
.text:0000DA66          cmp    dl, 0Bh
.text:0000DA69          jnz   short loc_DA83
.text:0000DA6B          cmp    esi, 0FE81h
.text:0000DA71          jz    OK
.text:0000DA77          cmp    esi, 0FFFFF8EFh
.text:0000DA7D          jz    OK
.text:0000DA83          ; CODE 
    ↳ XREF: sync_sys+201j
.text:0000DA83 loc_DA83:
.text:0000DA83          mov    cl, [ebx]
.text:0000DA85          cmp    cl, 0Ch
.text:0000DA88          jnz   short loc_DA9F
.text:0000DA8A          cmp    esi, 12A9h
.text:0000DA90          jz    OK
.text:0000DA96          cmp    esi, 0FFFFFF5h
.text:0000DA99          jz    OK
.text:0000DA9F

```

```

.text:0000DA9F loc_DA9F: ; CODE 
    ↳ XREF: sync_sys+220j
.text:0000DA9F          mov    eax, [ebp+var_18]
.text:0000DAA2          test   eax, eax
.text:0000DAA4          jz    short loc_DAB0
.text:0000DAA6          push   24h
.text:0000DAA8          call   timer
.text:0000DAAD          add    esp, 4
.text:0000DAB0
.text:0000DAB0 loc_DAB0: ; CODE 
    ↳ XREF: sync_sys+23Cj
.text:0000DAB0          inc    edi
.text:0000DAB1          cmp    edi, 3
.text:0000DAB4          jle    short loc_DA55
.text:0000DAB6          mov    eax, ds:net_env
.text:0000DABB          test   eax, eax
.text:0000DABD          jz    short error

...
.text:0000DAF7 error: ; CODE 
    ↳ XREF: sync_sys+255j
.text:0000DAF7          ; 
    ↳ sync_sys+274j ...
.text:0000DAF7          mov    [ebp+var_8], offset 
    ↳ encrypted_error_message2
.text:0000DAFE          mov    [ebp+var_C], 17h ; 
    ↳ decrypting key
.text:0000DB05          jmp    decrypt_end_print_message

...
; this name we gave to label:
.text:0000D9B6 decrypt_end_print_message: ; CODE 
    ↳ XREF: sync_sys+29Dj
.text:0000D9B6          ; 
    ↳ sync_sys+2ABj
.text:0000D9B6          mov    eax, [ebp+var_18]
.text:0000D9B9          test   eax, eax
.text:0000D9BB          jnz   short loc_D9FB
.text:0000D9BD          mov    edx, [ebp+var_C] ; key
.text:0000D9C0          mov    ecx, [ebp+var_8] ; 
    ↳ string
.text:0000D9C3          push   edx
.text:0000D9C4          push   20h
.text:0000D9C6          push   ecx

```

```

.text:0000D9C7          push    18h
.text:0000D9C9          call    err_warn
.text:0000D9CE          push    0Fh
.text:0000D9D0          push    190h
.text:0000D9D5          call    sound
.text:0000D9DA          mov     [ebp+var_18], 1
.text:0000D9E1          add    esp, 18h
.text:0000D9E4          call    pcv_kbhit
.text:0000D9E9          test   eax, eax
.text:0000D9EB          jz     short loc_D9FB

...
; this name we gave to label:
.data:00401736 encrypted_error_message2 db 74h, 72h, 78h, 43h, ↴
    ↴ 48h, 6, 5Ah, 49h, 4Ch, 2 dup(47h)
.data:00401736           db 51h, 4Fh, 47h, 61h, 20h, 22h, ↴
    ↴ 3Ch, 24h, 33h, 36h, 76h
.data:00401736           db 3Ah, 33h, 31h, 0Ch, 0, 0Bh, 1 ↴
    ↴ Fh, 7, 1Eh, 1Ah

```

78.2.1

. err_warn() :

Listing 78.1:

```

.text:0000A44D          mov     esi, [ebp+arg_C] ; key
.text:0000A450          mov     edx, [ebp+arg_4] ; ↴
    ↴ string
.text:0000A453 loc_A453:
.text:0000A453          xor    eax, eax
.text:0000A455          mov     al, [edx+edi] ;
.text:0000A458          xor    eax, esi ;
.text:0000A45A          add    esi, 3 ;
.text:0000A45D          inc    edi
.text:0000A45E          cmp    edi, ecx
.text:0000A460          mov     [ebp+edi+var_55], al
.text:0000A464          jl     short loc_A453

```

:

```

.text:0000DAF7 error: ; CODE 
    ↴ XREF: sync_sys+255j
.text:0000DAF7 ; 
    ↴ sync_sys+274j ...
.text:0000DAF7         mov     [ebp+var_8], offset 
    ↴ encrypted_error_message2
.text:0000DAFE         mov     [ebp+var_C], 17h ; 
    ↴ decrypting key
.text:0000DB05         jmp     decrypt_end_print_message

...
; this name we gave to label manually:
.text:0000D9B6 decrypt_end_print_message: ; CODE 
    ↴ XREF: sync_sys+29Dj
.text:0000D9B6 ; 
    ↴ sync_sys+2ABj
.text:0000D9B6         mov     eax, [ebp+var_18]
.text:0000D9B9         test    eax, eax
.text:0000D9BB         jnz    short loc_D9FB
.text:0000D9BD         mov     edx, [ebp+var_C] ; key
.text:0000D9C0         mov     ecx, [ebp+var_8] ; 
    ↴ string
.text:0000D9C3         push    edx
.text:0000D9C4         push    20h
.text:0000D9C6         push    ecx
.text:0000D9C7         push    18h
.text:0000D9C9         call    err_warn

```

xoring: .

:

Listing 78.2: Python 3.x

```

#!/usr/bin/python
import sys

msg=[0x74, 0x72, 0x78, 0x43, 0x48, 0x6, 0x5A, 0x49, 0x4C, 0x47, 
    ↴ 0x47,
0x51, 0x4F, 0x47, 0x61, 0x20, 0x22, 0x3C, 0x24, 0x33, 0x36, 0x3A, 
    ↴ x76,
0x33, 0x31, 0x0C, 0x0, 0x0B, 0x1F, 0x7, 0x1E, 0x1A]

key=0x17
tmp=key
for i in msg:

```

```

    sys.stdout.write ("%c" % (i^tmp))
    tmp=tmp+3
sys.stdout.flush()

```

: «check security device connection»..

... (XOR). ESI. 255, .

0..255..

Listing 78.3: Python 3.x

```

#!/usr/bin/python
import sys, curses.ascii

msgs=[

[0x74, 0x72, 0x78, 0x43, 0x48, 0x6, 0x5A, 0x49, 0x4C, 0x47, 0x2F,
 ↴ x47,
0x51, 0x4F, 0x47, 0x61, 0x20, 0x22, 0x3C, 0x24, 0x33, 0x36, 0x2F,
 ↴ x76,
0x3A, 0x33, 0x31, 0x0C, 0x0, 0x0B, 0x1F, 0x7, 0x1E, 0x1A],

[0x49, 0x65, 0x2D, 0x63, 0x76, 0x75, 0x6C, 0x6E, 0x76, 0x56, 0x2F,
 ↴ x5C,
8, 0x4F, 0x4B, 0x47, 0x5D, 0x54, 0x5F, 0x1D, 0x26, 0x2C, 0x33,
0x27, 0x28, 0x6F, 0x72, 0x75, 0x78, 0x7B, 0x7E, 0x41, 0x44],

[0x45, 0x61, 0x31, 0x67, 0x72, 0x79, 0x68, 0x52, 0x4A, 0x52, 0x2F,
 ↴ x50,
0x0C, 0x4B, 0x57, 0x43, 0x51, 0x58, 0x5B, 0x61, 0x37, 0x33, 0x2F,
 ↴ x2B,
0x39, 0x39, 0x3C, 0x38, 0x79, 0x3A, 0x30, 0x17, 0x0B, 0x0C],

[0x40, 0x64, 0x79, 0x75, 0x7F, 0x6F, 0x0, 0x4C, 0x40, 0x9, 0x4D2F,
 ↴ , 0x5A,
0x46, 0x5D, 0x57, 0x49, 0x57, 0x3B, 0x21, 0x23, 0x6A, 0x38, 0x2F,
 ↴ x23,
0x36, 0x24, 0x2A, 0x7C, 0x3A, 0x1A, 0x6, 0x0D, 0x0E, 0x0A, 0x142F
 ↴ ,
0x10],

[0x72, 0x7C, 0x72, 0x79, 0x76, 0x0,
0x50, 0x43, 0x4A, 0x59, 0x5D, 0x5B, 0x41, 0x41, 0x1B, 0x5A,
0x24, 0x32, 0x2E, 0x29, 0x28, 0x70, 0x20, 0x22, 0x38, 0x28, 0x2F,
 ↴ x36,
0x0D, 0x0B, 0x48, 0x4B, 0x4E]]


def is_string_printable(s):

```

```

    return all(list(map(lambda x: curses.ascii.isprint(x), s)))

cnt=1
for msg in msgs:
    print ("message #%d" % cnt)
    for key in range(0,256):
        result=[]
        tmp=key
        for i in msg:
            result.append (i^tmp)
            tmp=tmp+3
        if is_string_printable (result):
            print ("key=", key, "value=", "".join(
                list(map(chr, result))))
    cnt=cnt+1

```

:

Listing 78.4: Results

```

message #1
key= 20 value= `eb^h%| ``hudw|_af{n~f%ljmSbnwlpk
key= 21 value= ajc]i"}cawtgv{^bgto}g"millcmvkqh
key= 22 value= bkd\j#rbbvsfuz!cduh|d#bhomdlujni
key= 23 value= check security device connection
key= 24 value= lifbl!pd|tqhsx#ejwjbb!`nQfbshlo
message #2
key= 7 value= No security device found
key= 8 value= An#rbbvsVuz!cduhld#gghtme?!#!'#!#
message #3
key= 7 value= Bk<waoqNUpu$`yreoa\wpmpusj,bkIjh
key= 8 value= Mj?vfnr0jqv%gxqd``_vwlstlk/clHii
key= 9 value= Lm>ugasLkvw&fgpgag^uvcrwml.`mwhj
key= 10 value= 0l!td`tMhwx'efwfbf!tubuvnm!anvok
key= 11 value= No security device station found
key= 12 value= In#rjbvsnuz!{duhdd#r{`whho#gPtme
message #4
key= 14 value= Number of authorized users exceeded
key= 15 value= Ovlmdq!hg#`juknuhydk!vrbsp!Zy`dbe
message #5
key= 17 value= check security device station
key= 18 value= `ijbh!td`tmhwx'efwfbf!tubuVnm!'!

```

!

..

78.3 #3: MS-DOS

MS-DOS 1995 .

. .
. .:
. .:

```
seg030:0034          out_port proc far ; CODE XREF: sent_pro+2Ap ...
    ↴ +22p
seg030:0034          ; sent_pro+2Ap ...
seg030:0034
seg030:0034          arg_0     = byte ptr  6
seg030:0034
seg030:0034 55        push      bp
seg030:0035 8B EC      mov       bp, sp
seg030:0037 8B 16 7E E7 mov       dx, _out_port ; 0x378
seg030:003B 8A 46 06      mov       al, [bp+arg_0]
seg030:003E EE        out      dx, al
seg030:003F 5D        pop       bp
seg030:0040 CB        retf
seg030:0040          out_port endp
```

()

out_port() :

```
seg030:0041          sent_pro proc far ; CODE XREF: ↴
    ↴ check_dongle+34p
seg030:0041
seg030:0041          var_3     = byte ptr -3
seg030:0041          var_2     = word ptr -2
seg030:0041          arg_0     = dword ptr  6
seg030:0041
seg030:0041 C8 04 00 00      enter    4, 0
seg030:0045 56        push      si
seg030:0046 57        push      di
seg030:0047 8B 16 82 E7      mov       dx, _in_port_1 ; 0x37A
seg030:004B EC        in       al, dx
seg030:004C 8A D8      mov       bl, al
seg030:004E 80 E3 FE      and      bl, 0FEh
seg030:0051 80 CB 04      or       bl, 4
seg030:0054 8A C3      mov       al, bl
seg030:0056 88 46 FD      mov       [bp+var_3], al
seg030:0059 80 E3 1F      and      bl, 1Fh
seg030:005C 8A C3      mov       al, bl
seg030:005E EE        out      dx, al
```

seg030:005F	68 FF 00	push	0FFh
seg030:0062	0E	push	cs
seg030:0063	E8 CE FF	call	near ptr out_port
seg030:0066	59	pop	cx
seg030:0067	68 D3 00	push	0D3h
seg030:006A	0E	push	cs
seg030:006B	E8 C6 FF	call	near ptr out_port
seg030:006E	59	pop	cx
seg030:006F	33 F6	xor	si, si
seg030:0071	EB 01	jmp	short loc_359D4
seg030:0073			
seg030:0073		loc_359D3: ; CODE XREF: sent_pro+37j	
seg030:0073	46	inc	si
seg030:0074			
seg030:0074		loc_359D4: ; CODE XREF: sent_pro+30j	
seg030:0074	81 FE 96 00	cmp	si, 96h
seg030:0078	7C F9	jl	short loc_359D3
seg030:007A	68 C3 00	push	0C3h
seg030:007D	0E	push	cs
seg030:007E	E8 B3 FF	call	near ptr out_port
seg030:0081	59	pop	cx
seg030:0082	68 C7 00	push	0C7h
seg030:0085	0E	push	cs
seg030:0086	E8 AB FF	call	near ptr out_port
seg030:0089	59	pop	cx
seg030:008A	68 D3 00	push	0D3h
seg030:008D	0E	push	cs
seg030:008E	E8 A3 FF	call	near ptr out_port
seg030:0091	59	pop	cx
seg030:0092	68 C3 00	push	0C3h
seg030:0095	0E	push	cs
seg030:0096	E8 9B FF	call	near ptr out_port
seg030:0099	59	pop	cx
seg030:009A	68 C7 00	push	0C7h
seg030:009D	0E	push	cs
seg030:009E	E8 93 FF	call	near ptr out_port
seg030:00A1	59	pop	cx
seg030:00A2	68 D3 00	push	0D3h
seg030:00A5	0E	push	cs
seg030:00A6	E8 8B FF	call	near ptr out_port
seg030:00A9	59	pop	cx
seg030:00AA	BF FF FF	mov	di, 0FFFFh
seg030:00AD	EB 40	jmp	short loc_35A4F
seg030:00AF			
seg030:00AF		loc_35A0F: ; CODE XREF: sent_pro+BDj	
seg030:00AF	BE 04 00	mov	si, 4
seg030:00B2			

seg030:00B2	loc_35A12: ; CODE XREF: sent_pro+ACj
seg030:00B2 D1 E7	shl di, 1
seg030:00B4 8B 16 80 E7	mov dx, _in_port_2 ; 0x379
seg030:00B8 EC	in al, dx
seg030:00B9 A8 80	test al, 80h
seg030:00BB 75 03	jnz short loc_35A20
seg030:00BD 83 CF 01	or di, 1
seg030:00C0	
seg030:00C0	loc_35A20: ; CODE XREF: sent_pro+7Aj
seg030:00C0 F7 46 FE 08+	test [bp+var_2], 8
seg030:00C5 74 05	jz short loc_35A2C
seg030:00C7 68 D7 00	push 0D7h ; '+'
seg030:00CA EB 0B	jmp short loc_35A37
seg030:00CC	
seg030:00CC	loc_35A2C: ; CODE XREF: sent_pro+84j
seg030:00CC 68 C3 00	push 0C3h
seg030:00CF 0E	push cs
seg030:00D0 E8 61 FF	call near ptr out_port
seg030:00D3 59	pop cx
seg030:00D4 68 C7 00	push 0C7h
seg030:00D7	
seg030:00D7	loc_35A37: ; CODE XREF: sent_pro+89j
seg030:00D7 0E	push cs
seg030:00D8 E8 59 FF	call near ptr out_port
seg030:00DB 59	pop cx
seg030:00DC 68 D3 00	push 0D3h
seg030:00DF 0E	push cs
seg030:00E0 E8 51 FF	call near ptr out_port
seg030:00E3 59	pop cx
seg030:00E4 8B 46 FE	mov ax, [bp+var_2]
seg030:00E7 D1 E0	shl ax, 1
seg030:00E9 89 46 FE	mov [bp+var_2], ax
seg030:00EC 4E	dec si
seg030:00ED 75 C3	jnz short loc_35A12
seg030:00EF	
seg030:00EF	loc_35A4F: ; CODE XREF: sent_pro+6Cj
seg030:00EF C4 5E 06	les bx, [bp+arg_0]
seg030:00F2 FF 46 06	inc word ptr [bp+arg_0]
seg030:00F5 26 8A 07	mov al, es:[bx]
seg030:00F8 98	cbw
seg030:00F9 89 46 FE	mov [bp+var_2], ax
seg030:00FC 0B C0	or ax, ax
seg030:00FE 75 AF	jnz short loc_35A0F
seg030:0100 68 FF 00	push 0FFh
seg030:0103 0E	push cs
seg030:0104 E8 2D FF	call near ptr out_port
seg030:0107 59	pop cx

seg030:0108	8B 16 82 E7	mov	dx, _in_port_1 ; 0x37A
seg030:010C	EC	in	al, dx
seg030:010D	8A C8	mov	cl, al
seg030:010F	80 E1 5F	and	cl, 5Fh
seg030:0112	8A C1	mov	al, cl
seg030:0114	EE	out	dx, al
seg030:0115	EC	in	al, dx
seg030:0116	8A C8	mov	cl, al
seg030:0118	F6 C1 20	test	cl, 20h
seg030:011B	74 08	jz	short loc_35A85
seg030:011D	8A 5E FD	mov	bl, [bp+var_3]
seg030:0120	80 E3 DF	and	bl, 0DFh
seg030:0123	EB 03	jmp	short loc_35A88
seg030:0125			
seg030:0125		loc_35A85: ; CODE XREF: sent_pro+DAj	
seg030:0125	8A 5E FD	mov	bl, [bp+var_3]
seg030:0128			
seg030:0128		loc_35A88: ; CODE XREF: sent_pro+E2j	
seg030:0128	F6 C1 80	test	cl, 80h
seg030:012B	74 03	jz	short loc_35A90
seg030:012D	80 E3 7F	and	bl, 7Fh
seg030:0130			
seg030:0130		loc_35A90: ; CODE XREF: sent_pro+EAj	
seg030:0130	8B 16 82 E7	mov	dx, _in_port_1 ; 0x37A
seg030:0134	8A C3	mov	al, bl
seg030:0136	EE	out	dx, al
seg030:0137	8B C7	mov	ax, di
seg030:0139	5F	pop	di
seg030:013A	5E	pop	si
seg030:013B	C9	leave	
seg030:013C	CB	retf	
seg030:013C		sent_pro	endp

..

.. 5 ..

_in_port_2 (0x379) y _in_port_1 (0x37A).

seg030:00B9: .

? .

:

00000000	struct_0	struc ; (sizeof=0x1B)
00000000	field_0	db 25 dup(?) ; string(C)
00000019	_A	dw ?

```

0000001B struct_0      ends

dseg:3CBC 61 63 72 75+_Q    struct_0 <'hello', 01122h>
dseg:3CBC 6E 00 00 00+      ; DATA XREF: check_dongle+2Eo

... skipped ...

dseg:3E00 63 6F 66 66+      struct_0 <'coffee', 7EB7h>
dseg:3E1B 64 6F 67 00+      struct_0 <'dog', OFFADh>
dseg:3E36 63 61 74 00+      struct_0 <'cat', OFF5Fh>
dseg:3E51 70 61 70 65+      struct_0 <'paper', OFFDFh>
dseg:3E6C 63 6F 6B 65+      struct_0 <'coke', 0F568h>
dseg:3E87 63 6C 6F 63+      struct_0 <'clock', 55EAh>
dseg:3EA2 64 69 72 00+      struct_0 <'dir', OFFAEh>
dseg:3EBD 63 6F 70 79+      struct_0 <'copy', 0F557h>

seg030:0145          check_dongle proc far ; CODE XREF: ↵
    ↓ sub_3771D+3EP
seg030:0145
seg030:0145          var_6 = dword ptr -6
seg030:0145          var_2 = word ptr -2
seg030:0145
seg030:0145 C8 06 00 00      enter   6, 0
seg030:0149 56          push     si
seg030:014A 66 6A 00      push     large 0           ; newtime
seg030:014D 6A 00          push     0                 ; cmd
seg030:014F 9A C1 18 00+    call    _biostime
seg030:0154 52          push     dx
seg030:0155 50          push     ax
seg030:0156 66 58          pop     eax
seg030:0158 83 C4 06          add    sp, 6
seg030:015B 66 89 46 FA      mov    [bp+var_6], eax
seg030:015F 66 3B 06 D8+    cmp    eax, _expiration
seg030:0164 7E 44          jle    short loc_35B0A
seg030:0166 6A 14          push   14h
seg030:0168 90          nop
seg030:0169 0E          push   cs
seg030:016A E8 52 00      call   near ptr get_rand
seg030:016D 59          pop    cx
seg030:016E 8B F0          mov    si, ax
seg030:0170 6B C0 1B      imul  ax, 1Bh
seg030:0173 05 BC 3C      add   ax, offset _Q
seg030:0176 1E          push   ds
seg030:0177 50          push   ax
seg030:0178 0E          push   cs
seg030:0179 E8 C5 FE      call   near ptr sent_pro

```

seg030:017C	83 C4 04	add	sp, 4
seg030:017F	89 46 FE	mov	[bp+var_2], ax
seg030:0182	8B C6	mov	ax, si
seg030:0184	6B C0 12	imul	ax, 18
seg030:0187	66 0F BF C0	movsx	eax, ax
seg030:018B	66 8B 56 FA	mov	edx, [bp+var_6]
seg030:018F	66 03 D0	add	edx, eax
seg030:0192	66 89 16 D8+	mov	_expiration, edx
seg030:0197	8B DE	mov	bx, si
seg030:0199	6B DB 1B	imul	bx, 27
seg030:019C	8B 87 D5 3C	mov	ax, _Q._A[bx]
seg030:01A0	3B 46 FE	cmp	ax, [bp+var_2]
seg030:01A3	74 05	jz	short loc_35B0A
seg030:01A5	B8 01 00	mov	ax, 1
seg030:01A8	EB 02	jmp	short loc_35B0C
seg030:01AA			
seg030:01AA		loc_35B0A:	; CODE XREF: check_dongle+1 ↴
	↳ Fj		
seg030:01AA			; check_dongle+5Ej
seg030:01AA	33 C0	xor	ax, ax
seg030:01AC			
seg030:01AC		loc_35B0C:	; CODE XREF: check_dongle+63 ↴
	↳ j		
seg030:01AC	5E	pop	si
seg030:01AD	C9	leave	
seg030:01AE	CB	retf	
seg030:01AE		check_dongle	endp

get_rand() :

seg030:01BF		get_rand	proc far ; CODE XREF: ↴
	↳ check_dongle+25p		
seg030:01BF		arg_0	= word ptr 6
seg030:01BF			
seg030:01BF	55	push	bp
seg030:01C0	8B EC	mov	bp, sp
seg030:01C2	9A 3D 21 00+	call	_rand
seg030:01C7	66 0F BF C0	movsx	eax, ax
seg030:01CB	66 0F BF 56+	movsx	edx, [bp+arg_0]
seg030:01D0	66 0F AF C2	imul	eax, edx
seg030:01D4	66 BB 00 80+	mov	ebx, 8000h
seg030:01DA	66 99	cdq	
seg030:01DC	66 F7 FB	idiv	ebx
seg030:01DF	5D	pop	bp
seg030:01E0	CB	retf	

seg030:01E0	get_rand	endp
-------------	----------	------

```

.
.

seg033:087B 9A 45 01 96+    call    check_dongle
seg033:0880 0B C0            or      ax, ax
seg033:0882 74 62            jz     short OK
seg033:0884 83 3E 60 42+    cmp    word_620E0, 0
seg033:0889 75 5B            jnz    short OK
seg033:088B FF 06 60 42    inc    word_620E0
seg033:088F 1E              push   ds
seg033:0890 68 22 44        push   offset aTrupcRequiresA ; "↙
    ↳ This Software Requires a Software Lock\n"
seg033:0893 1E              push   ds
seg033:0894 68 60 E9        push   offset byte_6C7E0 ; dest
seg033:0897 9A 79 65 00+    call   _strcpy
seg033:089C 83 C4 08        add    sp, 8
seg033:089F 1E              push   ds
seg033:08A0 68 42 44        push   offset aPleaseContactA ; "↙
    ↳ Please Contact ..."
seg033:08A3 1E              push   ds
seg033:08A4 68 60 E9        push   offset byte_6C7E0 ; dest
seg033:08A7 9A CD 64 00+    call   _strcat

```

```

.
.

mov ax,0
retf

```

```

.
.

seg033:088F 1E                  push   ds
seg033:0890 68 22 44            push   offset ↵
    ↳ aTrupcRequiresA ; "This Software Requires a Software Lock↙
    ↳ \n"
seg033:0893 1E                  push   ds
seg033:0894 68 60 E9            push   offset ↵
    ↳ byte_6C7E0 ; dest
seg033:0897 9A 79 65 00+        call   _strcpy
seg033:089C 83 C4 08            add    sp, 8

```

: 94 on page 1160.

.

DS

.

Capítulo 79

```
:
.text:00541000 set_bit          proc near           ; CODE 
    ↳ XREF: rotate1+42
.text:00541000
    ↳ rotate2+42 ...
.text:00541000
.text:00541000 arg_0           = dword ptr  4
.text:00541000 arg_4           = dword ptr  8
.text:00541000 arg_8           = dword ptr  0Ch
.text:00541000 arg_C           = byte ptr   10h
.text:00541000
.text:00541000
    mov     al, [esp+arg_C]
.text:00541004
    mov     ecx, [esp+arg_8]
.text:00541008
    push    esi
.text:00541009
    mov     esi, [esp+4+arg_0]
.text:0054100D
    test    al, al
.text:0054100F
    mov     eax, [esp+4+arg_4]
.text:00541013
    mov     dl, 1
.text:00541015
    jz      short loc_54102B
.text:00541017
    shl     dl, cl
.text:00541019
    mov     cl, cube64[eax+esi*8]
.text:00541020
    or     cl, dl
.text:00541022
    mov     cube64[eax+esi*8], cl
.text:00541029
    pop     esi
.text:0054102A
    retn
.text:0054102B
.text:0054102B loc_54102B:        ; CODE 
    ↳ XREF: set_bit+15
.text:0054102B
    shl     dl, cl
.text:0054102D
    mov     cl, cube64[eax+esi*8]
.text:00541034
    not    dl
.text:00541036
    and    cl, dl
```

```

.text:00541038          mov     cube64[eax+esi*8], cl
.text:0054103F          pop     esi
.text:00541040          retn
.text:00541040 set_bit    endp
.text:00541040
.text:00541041          align 10h
.text:00541050
.text:00541050 ; ====== S U B R O U T I N E ↴
    ↴ ======
.text:00541050
.text:00541050
.text:00541050 get_bit    proc near             ; CODE ↴
    ↴ XREF: rotate1+16
.text:00541050
.text:00541050          ; rotate2+16 ...
.text:00541050
.text:00541050 arg_0      = dword ptr  4
.text:00541050 arg_4      = dword ptr  8
.text:00541050 arg_8      = byte ptr   0Ch
.text:00541050
.text:00541050          mov     eax, [esp+arg_4]
.text:00541054          mov     ecx, [esp+arg_0]
.text:00541058          mov     al, cube64[eax+ecx*8]
.text:0054105F          mov     cl, [esp+arg_8]
.text:00541063          shr     al, cl
.text:00541065          and     al, 1
.text:00541067          retn
.text:00541067 get_bit    endp
.text:00541067
.text:00541068          align 10h
.text:00541070
.text:00541070 ; ====== S U B R O U T I N E ↴
    ↴ ======
.text:00541070
.text:00541070
.text:00541070 rotate1     proc near             ; CODE ↴
    ↴ XREF: rotate_all_with_password+8E
.text:00541070
.text:00541070 internal_array_64= byte ptr -40h
.text:00541070 arg_0       = dword ptr  4
.text:00541070
.text:00541070          sub     esp, 40h
.text:00541073          push    ebx
.text:00541074          push    ebp
.text:00541075          mov     ebp, [esp+48h+arg_0]
.text:00541079          push    esi
.text:0054107A          push    edi

```

```

.text:0054107B          xor    edi, edi      ; EDI is ↴
    ↳ loop1 counter
.text:0054107D          lea    ebx, [esp+50h+ ↳
    ↳ internal_array_64]
.text:00541081
.text:00541081 first_loop1_begin:           ; CODE ↴
    ↳ XREF: rotate1+2E
.text:00541081 xor    esi, esi      ; ESI is ↴
    ↳ loop2 counter
.text:00541083
.text:00541083 first_loop2_begin:           ; CODE ↴
    ↳ XREF: rotate1+25
.text:00541083 push   ebp          ; arg_0
.text:00541084 push   esi
.text:00541085 push   edi
.text:00541086 call   get_bit
.text:0054108B add    esp, 0Ch
.text:0054108E mov    [ebx+esi], al  ; store ↴
    ↳ to internal array
.text:00541091 inc    esi
.text:00541092 cmp    esi, 8
.text:00541095 j1    short first_loop2_begin
.text:00541097 inc    edi
.text:00541098 add    ebx, 8
.text:0054109B cmp    edi, 8
.text:0054109E j1    short first_loop1_begin
.text:005410A0 lea    ebx, [esp+50h+ ↳
    ↳ internal_array_64]
.text:005410A4 mov    edi, 7      ; EDI is ↴
    ↳ loop1 counter, initial state is 7
.text:005410A9
.text:005410A9 second_loop1_begin:          ; CODE ↴
    ↳ XREF: rotate1+57
.text:005410A9 xor    esi, esi      ; ESI is ↴
    ↳ loop2 counter
.text:005410AB
.text:005410AB second_loop2_begin:          ; CODE ↴
    ↳ XREF: rotate1+4E
.text:005410AB mov    al, [ebx+esi]  ; value ↴
    ↳ from internal array
.text:005410AE push   eax
.text:005410AF push   ebp          ; arg_0
.text:005410B0 push   edi
.text:005410B1 push   esi
.text:005410B2 call   set_bit
.text:005410B7 add    esp, 10h
.text:005410BA inc    esi          ; ↴

```

```

    ↳ increment loop2 counter
.text:005410BB             cmp    esi, 8
.text:005410BE             jl     short second_loop2_begin
.text:005410C0             dec    edi           ; ↳
    ↳ decrement loop2 counter
.text:005410C1             add    ebx, 8
.text:005410C4             cmp    edi, 0FFFFFFFh
.text:005410C7             jg    short second_loop1_begin
.text:005410C9             pop    edi
.text:005410CA             pop    esi
.text:005410CB             pop    ebp
.text:005410CC             pop    ebx
.text:005410CD             add    esp, 40h
.text:005410D0             retn
.text:005410D0 rotate1      endp
.text:005410D0
.text:005410D1             align 10h
.text:005410E0
.text:005410E0 ; ===== S U B R O U T I N E ↳
    =====
.text:005410E0
.text:005410E0
.text:005410E0 rotate2      proc near ; CODE ↳
    ↳ XREF: rotate_all_with_password+7A
.text:005410E0
.text:005410E0 internal_array_64= byte ptr -40h
.text:005410E0 arg_0         = dword ptr 4
.text:005410E0
.text:005410E0             sub    esp, 40h
.text:005410E3             push   ebx
.text:005410E4             push   ebp
.text:005410E5             mov    ebp, [esp+48h+arg_0]
.text:005410E9             push   esi
.text:005410EA             push   edi
.text:005410EB             xor    edi, edi ; loop1 ↳
    ↳ counter
.text:005410ED             lea    ebx, [esp+50h+ ↳
    ↳ internal_array_64]
.text:005410F1
.text:005410F1 loc_5410F1: ; CODE ↳
    ↳ XREF: rotate2+2E
.text:005410F1             xor    esi, esi ; loop2 ↳
    ↳ counter
.text:005410F3
.text:005410F3 loc_5410F3: ; CODE ↳
    ↳ XREF: rotate2+25
.text:005410F3             push   esi ; loop2

```

```

.text:005410F4      push    edi          ; loop1
.text:005410F5      push    ebp          ; arg_0
.text:005410F6      call    get_bit
.text:005410FB      add     esp, 0Ch
.text:005410FE      mov     [ebx+esi], al   ; store ↴
    ↴ to internal array
.text:00541101      inc    esi          ; ↵
    ↴ increment loop1 counter
.text:00541102      cmp    esi, 8
.text:00541105      jl    short loc_5410F3
.text:00541107      inc    edi          ; ↵
    ↴ increment loop2 counter
.text:00541108      add    ebx, 8
.text:0054110B      cmp    edi, 8
.text:0054110E      jl    short loc_5410F1
.text:00541110      lea    ebx, [esp+50h+] ↴
    ↴ internal_array_64]
.text:00541114      mov    edi, 7       ; loop1 ↵
    ↴ counter is initial state 7
.text:00541119      ; CODE ↵
    ↴ XREF: rotate2+57
.text:00541119 loc_541119:           ; CODE ↵
    ↴ XREF: rotate2+4E
.text:00541119      xor    esi, esi    ; loop2 ↵
    ↴ counter
.text:0054111B      ; CODE ↵
    ↴ XREF: rotate2+4E
.text:0054111B      mov    al, [ebx+esi] ; get ↵
    ↴ byte from internal array
.text:0054111E      push   eax
.text:0054111F      push   edi          ; loop1 ↵
    ↴ counter
.text:00541120      push   esi          ; loop2 ↵
    ↴ counter
.text:00541121      push   ebp          ; arg_0
.text:00541122      call   set_bit
.text:00541127      add    esp, 10h
.text:0054112A      inc    esi          ; ↵
    ↴ increment loop2 counter
.text:0054112B      cmp    esi, 8
.text:0054112E      jl    short loc_54111B
.text:00541130      dec    edi          ; ↵
    ↴ decrement loop2 counter
.text:00541131      add    ebx, 8
.text:00541134      cmp    edi, 0FFFFFFFh
.text:00541137      jg    short loc_541119
.text:00541139      pop    edi

```

```

.text:0054113A          pop    esi
.text:0054113B          pop    ebp
.text:0054113C          pop    ebx
.text:0054113D          add    esp, 40h
.text:00541140          retn
.text:00541140 rotate2   endp
.text:00541140
.text:00541141          align 10h
.text:00541150
.text:00541150 ; ===== S U B R O U T I N E ↴
    ↴ =====
.text:00541150
.text:00541150
.text:00541150 rotate3      proc near           ; CODE ↴
    ↴ XREF: rotate_all_with_password+66
.text:00541150
.text:00541150 var_40       = byte ptr -40h
.text:00541150 arg_0        = dword ptr 4
.text:00541150
.text:00541150             sub    esp, 40h
.text:00541153             push   ebx
.text:00541154             push   ebp
.text:00541155             mov    ebp, [esp+48h+arg_0]
.text:00541159             push   esi
.text:0054115A             push   edi
.text:0054115B             xor    edi, edi
.text:0054115D             lea    ebx, [esp+50h+var_40]
.text:00541161
.text:00541161 loc_541161: ; CODE ↴
    ↴ XREF: rotate3+2E
.text:00541161             xor    esi, esi
.text:00541163
.text:00541163 loc_541163: ; CODE ↴
    ↴ XREF: rotate3+25
.text:00541163             push   esi
.text:00541164             push   ebp
.text:00541165             push   edi
.text:00541166             call   get_bit
.text:0054116B             add    esp, 0Ch
.text:0054116E             mov    [ebx+esi], al
.text:00541171             inc    esi
.text:00541172             cmp    esi, 8
.text:00541175             jl    short loc_541163
.text:00541177             inc    edi
.text:00541178             add    ebx, 8
.text:0054117B             cmp    edi, 8
.text:0054117E             jl    short loc_541161

```

```

.text:00541180          xor    ebx, ebx
.text:00541182          lea    edi, [esp+50h+var_40]
.text:00541186          ; CODE 
    ↳ XREF: rotate3+54
.text:00541186 loc_541186:      ; CODE 
    ↳ XREF: rotate3+54
.text:00541186          mov    esi, 7
.text:0054118B          ; CODE 
    ↳ XREF: rotate3+4E
.text:0054118B          mov    al, [edi]
.text:0054118D          push   eax
.text:0054118E          push   ebx
.text:0054118F          push   ebp
.text:00541190          push   esi
.text:00541191          call   set_bit
.text:00541196          add    esp, 10h
.text:00541199          inc    edi
.text:0054119A          dec    esi
.text:0054119B          cmp    esi, OFFFFFFFFh
.text:0054119E          jg    short loc_54118B
.text:005411A0          inc    ebx
.text:005411A1          cmp    ebx, 8
.text:005411A4          jl    short loc_541186
.text:005411A6          pop    edi
.text:005411A7          pop    esi
.text:005411A8          pop    ebp
.text:005411A9          pop    ebx
.text:005411AA          add    esp, 40h
.text:005411AD          retn
.text:005411AD rotate3      endp
.text:005411AD
.text:005411AE          align 10h
.text:005411B0          ; ===== S U B R O U T I N E 
    ↳ =====
.text:005411B0
.text:005411B0          ; CODE 
    ↳ XREF: crypt+1F
.text:005411B0 rotate_all_with_password proc near      ; CODE 
    ↳ decrypt+36
.text:005411B0
.text:005411B0 arg_0        = dword ptr 4
.text:005411B0 arg_4        = dword ptr 8
.text:005411B0
.text:005411B0          mov    eax, [esp+arg_0]
.text:005411B4          push   ebp

```

```

.text:005411B5          mov    ebp, eax
.text:005411B7          cmp    byte ptr [eax], 0
.text:005411BA          jz    exit
.text:005411C0          push   ebx
.text:005411C1          mov    ebx, [esp+8+arg_4]
.text:005411C5          push   esi
.text:005411C6          push   edi
.text:005411C7          push   edi

.text:005411C7 loop_begin:                                ; CODE 
    ↳ XREF: rotate_all_with_password+9F
.text:005411C7          movsx  eax, byte ptr [ebp+0]
.text:005411CB          push   eax                         ; C
.text:005411CC          call   _tolower
.text:005411D1          add    esp, 4
.text:005411D4          cmp    al, 'a'
.text:005411D6          jl    short ↴
    ↳ next_character_in_password
.text:005411D8          cmp    al, 'z'
.text:005411DA          jg    short ↴
    ↳ next_character_in_password
.text:005411DC          movsx  ecx, al
.text:005411DF          sub    ecx, 'a'
.text:005411E2          cmp    ecx, 24
.text:005411E5          jle    short skip_subtracting
.text:005411E7          sub    ecx, 24
.text:005411EA          push   edi

.text:005411EA skip_subtracting:                          ; CODE 
    ↳ XREF: rotate_all_with_password+35
.text:005411EA          mov    eax, 55555556h
.text:005411EF          imul  ecx
.text:005411F1          mov    eax, edx
.text:005411F3          shr    eax, 1Fh
.text:005411F6          add    edx, eax
.text:005411F8          mov    eax, ecx
.text:005411FA          mov    esi, edx
.text:005411FC          mov    ecx, 3
.text:00541201          cdq
.text:00541202          idiv  ecx
.text:00541204          sub    edx, 0
.text:00541207          jz    short call_rotate1
.text:00541209          dec    edx
.text:0054120A          jz    short call_rotate2
.text:0054120C          dec    edx
.text:0054120D          jnz   short ↴
    ↳ next_character_in_password
.text:0054120F          test   ebx, ebx
.text:00541211          jle   short ↴

```

```

    ↳ next_character_in_password
.text:00541213          mov      edi, ebx
.text:00541215          call     rotate3; CODE ↴
    ↳ XREF: rotate_all_with_password+6F
.text:00541215          push    esi
.text:00541216          call    rotate3
.text:0054121B          add     esp, 4
.text:0054121E          dec     edi
.text:0054121F          jnz    short call_rotate3
.text:00541221          jmp     short ↵
    ↳ next_character_in_password
.text:00541223
.text:00541223 call_rotate2; CODE ↴
    ↳ XREF: rotate_all_with_password+5A
.text:00541223          test    ebx, ebx
.text:00541225          jle     short ↵
    ↳ next_character_in_password
.text:00541227          mov     edi, ebx
.text:00541229
.text:00541229 loc_541229; CODE ↴
    ↳ XREF: rotate_all_with_password+83
.text:00541229          push    esi
.text:0054122A          call    rotate2
.text:0054122F          add     esp, 4
.text:00541232          dec     edi
.text:00541233          jnz    short loc_541229
.text:00541235          jmp     short ↵
    ↳ next_character_in_password
.text:00541237
.text:00541237 call_rotate1; CODE ↴
    ↳ XREF: rotate_all_with_password+57
.text:00541237          test    ebx, ebx
.text:00541239          jle     short ↵
    ↳ next_character_in_password
.text:0054123B          mov     edi, ebx
.text:0054123D
.text:0054123D loc_54123D; CODE ↴
    ↳ XREF: rotate_all_with_password+97
.text:0054123D          push    esi
.text:0054123E          call    rotate1
.text:00541243          add     esp, 4
.text:00541246          dec     edi
.text:00541247          jnz    short loc_54123D
.text:00541249 next_character_in_password; CODE ↴
    ↳ XREF: rotate_all_with_password+26

```

```

.text:00541249          ; ↴
    ↳ rotate_all_with_password+2A ...
.text:00541249          mov     al, [ebp+1]
.text:0054124C          inc     ebp
.text:0054124D          test    al, al
.text:0054124F          jnz    loop_begin
.text:00541255          pop    edi
.text:00541256          pop    esi
.text:00541257          pop    ebx
.text:00541258          exit:               ; CODE ↴
    ↳ XREF: rotate_all_with_password+A
.text:00541258          pop    ebp
.text:00541259          retn
.text:00541259  rotate_all_with_password endp
.text:00541259
.text:0054125A          align 10h
.text:00541260
.text:00541260 ; ===== S U B R O U T I N E ↴
    ↳ =====
.text:00541260
.text:00541260
.text:00541260  crypt      proc near           ; CODE ↴
    ↳ XREF: crypt_file+8A
.text:00541260
.text:00541260  arg_0       = dword ptr 4
.text:00541260  arg_4       = dword ptr 8
.text:00541260  arg_8       = dword ptr 0Ch
.text:00541260
.text:00541260          push    ebx
.text:00541261          mov     ebx, [esp+4+arg_0]
.text:00541265          push    ebp
.text:00541266          push    esi
.text:00541267          push    edi
.text:00541268          xor    ebp, ebp
.text:0054126A
.text:0054126A loc_54126A:           ; CODE ↴
    ↳ XREF: crypt+41
.text:0054126A          mov     eax, [esp+10h+arg_8]
.text:0054126E          mov     ecx, 10h
.text:00541273          mov     esi, ebx
.text:00541275          mov     edi, offset cube64
.text:0054127A          push    1
.text:0054127C          push    eax
.text:0054127D          rep    movsd
.text:0054127F          call    rotate_all_with_password
.text:00541284          mov     eax, [esp+18h+arg_4]

```

```

.text:00541288          mov     edi, ebx
.text:0054128A          add     ebp, 40h
.text:0054128D          add     esp, 8
.text:00541290          mov     ecx, 10h
.text:00541295          mov     esi, offset cube64
.text:0054129A          add     ebx, 40h
.text:0054129D          cmp     ebp, eax
.text:0054129F          rep     movsd
.text:005412A1          jl    short loc_54126A
.text:005412A3          pop     edi
.text:005412A4          pop     esi
.text:005412A5          pop     ebp
.text:005412A6          pop     ebx
.text:005412A7          retn
.text:005412A7 crypt      endp
.text:005412A7
.align 10h
.text:005412B0 ; ====== S U B R O U T I N E ↴
↳ ======
.text:005412B0
.text:005412B0
.text:005412B0 ; int __cdecl decrypt(int, int, void *Src)
.text:005412B0 decrypt      proc near ; CODE ↴
↳ XREF: decrypt_file+99
.text:005412B0
.text:005412B0 arg_0        = dword ptr 4
.text:005412B0 arg_4        = dword ptr 8
.text:005412B0 Src          = dword ptr 0Ch
.text:005412B0
.text:005412B0             mov     eax, [esp+Src]
.text:005412B4             push    ebx
.text:005412B5             push    ebp
.text:005412B6             push    esi
.text:005412B7             push    edi
.text:005412B8             push    eax ; Src
.text:005412B9             call    __strupd
.text:005412BE             push    eax ; Str
.text:005412BF             mov     [esp+18h+Src], eax
.text:005412C3             call    __strrev
.text:005412C8             mov     ebx, [esp+18h+arg_0]
.text:005412CC             add     esp, 8
.text:005412CF             xor     ebp, ebp
.text:005412D1 loc_5412D1: ; CODE ↴
↳ XREF: decrypt+58
.text:005412D1             mov     ecx, 10h

```

```

.text:005412D6          mov     esi, ebx
.text:005412D8          mov     edi, offset cube64
.text:005412DD          push    3
.text:005412DF          rep     movsd
.text:005412E1          mov     ecx, [esp+14h+Src]
.text:005412E5          push    ecx
.text:005412E6          call    rotate_all_with_password
.text:005412EB          mov     eax, [esp+18h+arg_4]
.text:005412EF          mov     edi, ebx
.text:005412F1          add     ebp, 40h
.text:005412F4          add     esp, 8
.text:005412F7          mov     ecx, 10h
.text:005412FC          mov     esi, offset cube64
.text:00541301          add     ebx, 40h
.text:00541304          cmp     ebp, eax
.text:00541306          rep     movsd
.text:00541308          jl    short loc_5412D1
.text:0054130A          mov     edx, [esp+10h+Src]
.text:0054130E          push    edx           ; Memory
.text:0054130F          push    _free
.text:00541314          add     esp, 4
.text:00541317          pop     edi
.text:00541318          pop     esi
.text:00541319          pop     ebp
.text:0054131A          pop     ebx
.text:0054131B          retn
.text:0054131B decrypt  endp
.text:0054131B
.text:0054131C          align 10h
.text:00541320
.text:00541320 ; ===== S U B R O U T I N E ↴
    ↴ =====
.text:00541320
.text:00541320
.text:00541320 ; int __cdecl crypt_file(int Str, char *Filename, ↴
    ↴ , int password)
.text:00541320 crypt_file      proc near               ; CODE ↴
    ↴ XREF: _main+42
.text:00541320
.text:00541320 Str            = dword ptr 4
.text:00541320 Filename       = dword ptr 8
.text:00541320 password      = dword ptr 0Ch
.text:00541320
.text:00541320          mov     eax, [esp+Str]
.text:00541324          push    ebp
.text:00541325          push    offset Mode      ; "rb"
.text:0054132A          push    eax

```

```

    ↴ Filename
.text:0054132B      call   _fopen           ; open ↵
    ↴ file
.text:00541330      mov    ebp, eax
.text:00541332      add    esp, 8
.text:00541335      test   ebp, ebp
.text:00541337      jnz    short loc_541348
.text:00541339      push   offset Format    ; "↙
    ↴ Cannot open input file!\n"
.text:0054133E      call   _printf
.text:00541343      add    esp, 4
.text:00541346      pop    ebp
.text:00541347      retn
.text:00541348
.text:00541348 loc_541348:                      ; CODE ↵
    ↴ XREF: crypt_file+17
.text:00541348      push   ebx
.text:00541349      push   esi
.text:0054134A      push   edi
.text:0054134B      push   2                 ; Origin
.text:0054134D      push   0                 ; Offset
.text:0054134F      push   ebp
.text:00541350      call   _fseek
.text:00541355      push   ebp
.text:00541356      call   _ftell
    ↴ file size
.text:0054135B      push   0                 ; Origin
.text:0054135D      push   0                 ; Offset
.text:0054135F      push   ebp
.text:00541360      mov    [esp+2Ch+Str], eax
.text:00541364      call   _fseek
    ↴ to start
.text:00541369      mov    esi, [esp+2Ch+Str]
.text:0054136D      and   esi, 0FFFFFFC0h ; reset ↵
    ↴ all lowest 6 bits
.text:00541370      add   esi, 40h
    ↴ size to 64-byte border
.text:00541373      push   esi
.text:00541374      call   _malloc
.text:00541379      mov    ecx, esi
.text:0054137B      mov    ebx, eax
    ↴ allocated buffer pointer -> to EBX
.text:0054137D      mov    edx, ecx
.text:0054137F      xor    eax, eax
.text:00541381      mov    edi, ebx
.text:00541383      push   ebp
    ↴ File
.text:00541384      shr    ecx, 2

```

```

.text:00541387          rep stosd
.text:00541389          mov    ecx, edx
.text:0054138B          push   1           ; Count
.text:0054138D          and    ecx, 3
.text:00541390          rep stosb        ; memset
    ↳ (buffer, 0, aligned_size)
.text:00541392          mov    eax, [esp+38h+Str]
.text:00541396          push   eax         ; ↳
    ↳ ElementSize
.text:00541397          push   ebx         ; DstBuf
.text:00541398          call   _fread      ; read ↳
    ↳ file
.text:0054139D          push   ebp         ; File
.text:0054139E          call   _fclose     ; fclose
.text:005413A3          mov    ecx, [esp+44h+password]
.text:005413A7          push   ecx         ; ↳
    ↳ password
.text:005413A8          push   esi         ; ↳
    ↳ aligned size
.text:005413A9          push   ebx         ; buffer
.text:005413AA          call   crypt       ; do ↳
    ↳ crypt
.text:005413AF          mov    edx, [esp+50h+Filename]
.text:005413B3          add    esp, 40h
.text:005413B6          push   offset aWb     ; "wb"
.text:005413BB          push   edx         ; ↳
    ↳ Filename
.text:005413BC          call   _fopen      ; fopen
.text:005413C1          mov    edi, eax
.text:005413C3          push   edi         ; File
.text:005413C4          push   1           ; Count
.text:005413C6          push   3           ; Size
.text:005413C8          push   offset aQr9     ; "QR9"
.text:005413CD          call   _fwrite     ; write ↳
    ↳ file signature
.text:005413D2          push   edi         ; File
.text:005413D3          push   1           ; Count
.text:005413D5          lea    eax, [esp+30h+Str]
.text:005413D9          push   4           ; Size
.text:005413DB          push   eax         ; Str
.text:005413DC          call   _fwrite     ; write ↳
    ↳ original file size
.text:005413E1          push   edi         ; File
.text:005413E2          push   1           ; Count
.text:005413E4          push   esi         ; Size
.text:005413E5          push   ebx         ; Str
.text:005413E6          call   _fwrite     ; write ↳

```

```

↳ encrypted file
.text:005413EB      push    edi          ; File
.text:005413EC      call    _fclose
.text:005413F1      push    ebx          ; Memory
.text:005413F2      call    _free
.text:005413F7      add     esp, 40h
.text:005413FA      pop     edi
.text:005413FB      pop     esi
.text:005413FC      pop     ebx
.text:005413FD      pop     ebp
.text:005413FE      retn
.text:005413FE crypt_file    endp

.text:005413FE
.text:005413FF      align 10h
.text:00541400

.text:00541400 ; ===== S U B R O U T I N E =====
↳ =====
.text:00541400
.text:00541400
.text:00541400 ; int __cdecl decrypt_file(char *Filename, int, ↳
    ↳ void *Src)
.text:00541400 decrypt_file    proc near           ; CODE
    ↳ XREF: _main+6E
.text:00541400
.text:00541400 Filename        = dword ptr 4
.text:00541400 arg_4          = dword ptr 8
.text:00541400 Src            = dword ptr 0Ch
.text:00541400
.text:00541400             mov     eax, [esp+Filename]
.text:00541404      push    ebx
.text:00541405      push    ebp
.text:00541406      push    esi
.text:00541407      push    edi
.text:00541408      push    offset aRb      ; "rb"
.text:0054140D      push    eax
    ↳ Filename
.text:0054140E      call    _fopen
.text:00541413      mov     esi, eax
.text:00541415      add     esp, 8
.text:00541418      test   esi, esi
.text:0054141A      jnz    short loc_54142E
.text:0054141C      push    offset aCannotOpenIn_0 ; ↳
    ↳ "Cannot open input file!\n"
.text:00541421      call    _printf
.text:00541426      add     esp, 4
.text:00541429      pop     edi
.text:0054142A      pop     esi

```

```

.text:0054142B          pop    ebp
.text:0054142C          pop    ebx
.text:0054142D          retn
.text:0054142E
    ↳ XREF: decrypt_file+1A
.text:0054142E loc_54142E:           ; CODE 
.text:0054142E          push   2      ; Origin
.text:00541430          push   0      ; Offset
.text:00541432          push   esi    ; File
.text:00541433          call   _fseek
.text:00541438          push   esi    ; File
.text:00541439          call   _ftell
.text:0054143E          push   0      ; Origin
.text:00541440          push   0      ; Offset
.text:00541442          push   esi    ; File
.text:00541443          mov    ebp, eax
.text:00541445          call   _fseek
.text:0054144A          push   ebp    ; Size
.text:0054144B          call   _malloc
.text:00541450          push   esi    ; File
.text:00541451          mov    ebx, eax
.text:00541453          push   1      ; Count
.text:00541455          push   ebp    ; ↴
    ↳ ElementSize
.text:00541456          push   ebx    ; DstBuf
.text:00541457          call   _fread
.text:0054145C          push   esi    ; File
.text:0054145D          call   _fclose
.text:00541462          add    esp, 34h
.text:00541465          mov    ecx, 3
.text:0054146A          mov    edi, offset aQr9_0 ; " "
    ↳ QR9"
.text:0054146F          mov    esi, ebx
.text:00541471          xor    edx, edx
.text:00541473          repe  cmpsb
.text:00541475          jz    short loc_541489
.text:00541477          push   offset aFileIsNotCrypt ; ↴
    ↳ "File is not encrypted!\n"
.text:0054147C          call   _printf
.text:00541481          add    esp, 4
.text:00541484          pop    edi
.text:00541485          pop    esi
.text:00541486          pop    ebp
.text:00541487          pop    ebx
.text:00541488          retn
.text:00541489 loc_541489:           ; CODE 

```

```

    ↳ XREF: decrypt_file+75
.text:00541489          mov     eax, [esp+10h+Src]
.text:0054148D          mov     edi, [ebx+3]
.text:00541490          add     ebp, 0FFFFFFF9h
.text:00541493          lea     esi, [ebx+7]
.text:00541496          push    eax           ; Src
.text:00541497          push    ebp           ; int
.text:00541498          push    esi           ; int
.text:00541499          call    decrypt
.text:0054149E          mov     ecx, [esp+1Ch+arg_4]
.text:005414A2          push    offset awb_0   ; "wb"
.text:005414A7          push    ecx           ; ↳
    ↳ Filename
.text:005414A8          call    _fopen
.text:005414AD          mov     ebp, eax
.text:005414AF          push    ebp           ; File
.text:005414B0          push    1              ; Count
.text:005414B2          push    edi           ; Size
.text:005414B3          push    esi           ; Str
.text:005414B4          call    _fwrite
.text:005414B9          push    ebp           ; File
.text:005414BA          call    _fclose
.text:005414BF          push    ebx           ; Memory
.text:005414C0          call    _free
.text:005414C5          add     esp, 2Ch
.text:005414C8          pop    edi
.text:005414C9          pop    esi
.text:005414CA          pop    ebp
.text:005414CB          pop    ebx
.text:005414CC          retn
.text:005414CC decrypt_file    endp

```

```

.text:00541320 ; int __cdecl crypt_file(int Str, char *Filename,
    ↳ , int password)
.text:00541320 crypt_file      proc near
.text:00541320
.text:00541320 Str           = dword ptr  4
.text:00541320 Filename       = dword ptr  8
.text:00541320 password      = dword ptr  0Ch
.text:00541320

```

```

.text:00541320          mov     eax, [esp+Str]
.text:00541324          push    ebp
.text:00541325          push    offset Mode      ; "rb"
.text:0054132A          push    eax           ; ↳
    ↳ Filename

```

```
.text:0054132B           call    _fopen          ; open ↵
    ↴ file
.text:00541330           mov     ebp, eax
.text:00541332           add     esp, 8
.text:00541335           test    ebp, ebp
.text:00541337           jnz    short loc_541348
.text:00541339           push   offset Format    ; "↙
    ↴ Cannot open input file!\n"
.text:0054133E           call    _printf
.text:00541343           add     esp, 4
.text:00541346           pop    ebp
.text:00541347           retn
.text:00541348           loc_541348:
```

fseek()/ftell():

```
.text:00541348 push    ebx
.text:00541349 push    esi
.text:0054134A push    edi
.text:0054134B push    2      ; Origin
.text:0054134D push    0      ; Offset
.text:0054134F push    ebp    ; File

;
.text:00541350 call    _fseek
.text:00541355 push    ebp    ; File
.text:00541356 call    _ftell
.text:0054135B push    0      ; Origin
.text:0054135D push    0      ; Offset
.text:0054135F push    ebp    ; File
.text:00541360 mov     [esp+2Ch+Str], eax

;
.text:00541364 call    _fseek
```

```
.text:00541369 mov     esi, [esp+2Ch+Str]
;
.text:0054136D and     esi, 0FFFFFFC0h
;
.text:00541370 add     esi, 40h
```

.text:00541373	push	esi	Size
.text:00541374	call	_malloc	

```
.text:00541379 mov     ecx, esi
.text:0054137B mov     ebx, eax      ;
.text:0054137D mov     edx, ecx
.text:0054137F xor     eax, eax
.text:00541381 mov     edi, ebx
.text:00541383 push    ebp          ; File
.text:00541384 shr     ecx, 2
.text:00541387 rep stosd
.text:00541389 mov     ecx, edx
.text:0054138B push    1           ; Count
.text:0054138D and    ecx, 3
.text:00541390 rep stosb      ; memset (buffer, 0, )
```

`fread()`.

.text:00541392	mov	eax, [esp+38h+Str]
.text:00541396	push	eax ; ↴
↳ ElementSize		
.text:00541397	push	ebx ; DstBuf
.text:00541398	call	_fread ; read ↴
↳ file		
.text:0054139D	push	ebp ; File
.text:0054139E	call	_fclose ;

.text:005413A3	mov	ecx, [esp+44h+password]
.text:005413A7	push	ecx ; ↴
↳ password		
.text:005413A8	push	esi ; ↴
↳ aligned size		
.text:005413A9	push	ebx ; buffer
.text:005413AA	call	crypt ; do ↴
↳ crypt		

.text:005413AF	mov	edx, [esp+50h+Filename]
.text:005413B3	add	esp, 40h
.text:005413B6	push	offset awb ; "wb"
.text:005413BB	push	edx ; ↴
↳ Filename		
.text:005413BC	call	_fopen
.text:005413C1	mov	edi, eax

.text:005413C3	push	edi ; File
.text:005413C4	push	1 ; Count
.text:005413C6	push	3 ; Size
.text:005413C8	push	offset aQr9 ; "QR9"

.text:005413CD	call _fwrite	; write ↴
↳ file signature		

:

.text:005413D2	push edi	; File
.text:005413D3	push 1	; Count
.text:005413D5	lea eax, [esp+30h+Str]	
.text:005413D9	push 4	; Size
.text:005413DB	push eax	; Str
.text:005413DC	call _fwrite	; write ↴
↳ original file size		

:

.text:005413E1	push edi	; File
.text:005413E2	push 1	; Count
.text:005413E4	push esi	; Size
.text:005413E5	push ebx	; Str
.text:005413E6	call _fwrite	; write ↴
↳ encrypted file		

:

.text:005413EB	push edi	; File
.text:005413EC	call _fclose	
.text:005413F1	push ebx	; Memory
.text:005413F2	call _free	
.text:005413F7	add esp, 40h	
.text:005413FA	pop edi	
.text:005413FB	pop esi	
.text:005413FC	pop ebx	
.text:005413FD	pop ebp	
.text:005413FE	retn	
.text:005413FE crypt_file	endp	

:

```
void crypt_file(char *fin, char* fout, char *pw)
{
    FILE *f;
    int flen, flen_aligned;
    BYTE *buf;

    f=fopen(fin, "rb");

    if (f==NULL)
```

```

{
    printf ("Cannot open input file!\n");
    return;
};

fseek (f, 0, SEEK_END);
flen=f.tell (f);
fseek (f, 0, SEEK_SET);

flen_aligned=(flen&0xFFFFFC0)+0x40;

buf=(BYTE*)malloc (flen_aligned);
memset (buf, 0, flen_aligned);

fread (buf, flen, 1, f);

fclose (f);

crypt (buf, flen_aligned, pw);

f=fopen(fout, "wb");

fwrite ("QR9", 3, 1, f);
fwrite (&flen, 4, 1, f);
fwrite (buf, flen_aligned, 1, f);

fclose (f);

free (buf);
};

```

```

:
.text:00541400 ; int __cdecl decrypt_file(char *Filename, int, ↴
    ↴ void *Src)
.text:00541400 decrypt_file    proc near
.text:00541400
.text:00541400 Filename        = dword ptr  4
.text:00541400 arg_4          = dword ptr  8
.text:00541400 Src            = dword ptr  0Ch
.text:00541400
.text:00541400                 mov     eax, [esp+Filename]
.text:00541404                 push    ebx
.text:00541405                 push    ebp
.text:00541406                 push    esi
.text:00541407                 push    edi
.text:00541408                 push    offset aRb      ; "rb"

```

.text:0054140D	push	eax	; ↴
↳ Filename			
.text:0054140E	call	_fopen	
.text:00541413	mov	esi, eax	
.text:00541415	add	esp, 8	
.text:00541418	test	esi, esi	
.text:0054141A	jnz	short loc_54142E	
.text:0054141C	push	offset aCannotOpenIn_0 ; ↴	
↳ "Cannot open input file!\n"			
.text:00541421	call	_printf	
.text:00541426	add	esp, 4	
.text:00541429	pop	edi	
.text:0054142A	pop	esi	
.text:0054142B	pop	ebp	
.text:0054142C	pop	ebx	
.text:0054142D	retn		
.text:0054142E			
.text:0054142E loc_54142E:			
.text:0054142E	push	2	; Origin
.text:00541430	push	0	; Offset
.text:00541432	push	esi	; File
.text:00541433	call	_fseek	
.text:00541438	push	esi	; File
.text:00541439	call	_ftell	
.text:0054143E	push	0	; Origin
.text:00541440	push	0	; Offset
.text:00541442	push	esi	; File
.text:00541443	mov	ebp, eax	
.text:00541445	call	_fseek	
.text:0054144A	push	ebp	; Size
.text:0054144B	call	_malloc	
.text:00541450	push	esi	; File
.text:00541451	mov	ebx, eax	
.text:00541453	push	1	; Count
.text:00541455	push	ebp	; ↴
↳ ElementSize			
.text:00541456	push	ebx	; DstBuf
.text:00541457	call	_fread	
.text:0054145C	push	esi	; File
.text:0054145D	call	_fclose	

:

.text:00541462	add	esp, 34h	
.text:00541465	mov	ecx, 3	
.text:0054146A	mov	edi, offset aQr9_0 ; " ↴	
↳ QR9"			
.text:0054146F	mov	esi, ebx	

.text:00541471	xor	edx, edx
.text:00541473	repe	cmpsb
.text:00541475	jz	short loc_541489

:

.text:00541477	push	offset aFileIsNotCrypt ; ↴
↳ "File is not encrypted!\n"		
.text:0054147C	call	_printf
.text:00541481	add	esp, 4
.text:00541484	pop	edi
.text:00541485	pop	esi
.text:00541486	pop	ebp
.text:00541487	pop	ebx
.text:00541488	retn	
.text:00541489		
.text:00541489 loc_541489:		

decrypt().

.text:00541489	mov	eax, [esp+10h+Src]
.text:0054148D	mov	edi, [ebx+3]
.text:00541490	add	ebp, 0FFFFFF9h
.text:00541493	lea	esi, [ebx+7]
.text:00541496	push	eax ; Src
.text:00541497	push	ebp ; int
.text:00541498	push	esi ; int
.text:00541499	call	decrypt
.text:0054149E	mov	ecx, [esp+1Ch+arg_4]
.text:005414A2	push	offset awb_0 ; "wb"
.text:005414A7	push	ecx ; ↴
↳ Filename		
.text:005414A8	call	_fopen
.text:005414AD	mov	ebp, eax
.text:005414AF	push	ebp ; File
.text:005414B0	push	1 ; Count
.text:005414B2	push	edi ; Size
.text:005414B3	push	esi ; Str
.text:005414B4	call	_fwrite
.text:005414B9	push	ebp ; File
.text:005414BA	call	_fclose
.text:005414BF	push	ebx ; Memory
.text:005414C0	call	_free
.text:005414C5	add	esp, 2Ch
.text:005414C8	pop	edi
.text:005414C9	pop	esi
.text:005414CA	pop	ebp

```
.text:005414CB          pop     ebx
.text:005414CC          retn
.text:005414CC decrypt_file    endp
```

```
:
```

```
void decrypt_file(char *fin, char* fout, char *pw)
{
    FILE *f;
    int real_flen,flen;
    BYTE *buf;

    f=fopen(fin, "rb");

    if (f==NULL)
    {
        printf ("Cannot open input file!\n");
        return;
    };

    fseek (f, 0, SEEK_END);
    flen=f.tell (f);
    fseek (f, 0, SEEK_SET);

    buf=(BYTE*)malloc (flen);

    fread (buf, flen, 1, f);

    fclose (f);

    if (memcmp (buf, "QR9", 3)!=0)
    {
        printf ("File is not encrypted!\n");
        return;
    };

    memcpy (&real_flen, buf+3, 4);

    decrypt (buf+(3+4), flen-(3+4), pw);

    f=fopen(fout, "wb");

    fwrite (buf+(3+4), real_flen, 1, f);

    fclose (f);

    free (buf);
};
```

crypt():

```
.text:00541260 crypt          proc near
.text:00541260
.text:00541260 arg_0          = dword ptr 4
.text:00541260 arg_4          = dword ptr 8
.text:00541260 arg_8          = dword ptr 0Ch
.text:00541260
.text:00541260                 push    ebx
.text:00541261                 mov     ebx, [esp+4+arg_0]
.text:00541265                 push    ebp
.text:00541266                 push    esi
.text:00541267                 push    edi
.text:00541268                 xor    ebp, ebp
.text:0054126A loc_54126A:
```

```
.text:0054126A                 mov    eax, [esp+10h+arg_8]
.text:0054126E                 mov    ecx, 10h
.text:00541273                 mov    esi, ebx ; EBX is ↴
    ↴ pointer within input buffer
.text:00541275                 mov    edi, offset cube64
.text:0054127A                 push   1
.text:0054127C                 push   eax
.text:0054127D                 rep    movsd
```

rotate_all_with_password():

.text:0054127F	call	rotate_all_with_password
----------------	------	--------------------------

:

.text:00541284	mov	eax, [esp+18h+arg_4]
.text:00541288	mov	edi, ebx
.text:0054128A	add	ebp, 40h
.text:0054128D	add	esp, 8
.text:00541290	mov	ecx, 10h
.text:00541295	mov	esi, offset cube64
.text:0054129A	add	ebx, 40h ; add 64 to ↴ ↴ input buffer pointer
.text:0054129D	cmp	ebp, eax ; EBP contain ↴ ↴ amount of encrypted data.
.text:0054129F	rep	movsd

.text:005412A1	j1	short loc_54126A
.text:005412A3	pop	edi
.text:005412A4	pop	esi
.text:005412A5	pop	ebp
.text:005412A6	pop	ebx
.text:005412A7	retn	
.text:005412A7 crypt	endp	

:

```
void crypt (BYTE *buf, int sz, char *pw)
{
    int i=0;

    do
    {
        memcpy (cube, buf+i, 8*8);
        rotate_all (pw, 1);
        memcpy (buf+i, cube, 8*8);
        i+=64;
    }
    while (i<sz);
};
```

.text:005411B0	rotate_all_with_password	proc near
.text:005411B0		
.text:005411B0 arg_0	= dword ptr	4
.text:005411B0 arg_4	= dword ptr	8
.text:005411B0		
.text:005411B0	mov	eax, [esp+arg_0]
.text:005411B4	push	ebp
.text:005411B5	mov	ebp, eax

.text:005411B7	cmp	byte ptr [eax], 0
.text:005411BA	jz	exit
.text:005411C0	push	ebx
.text:005411C1	mov	ebx, [esp+8+arg_4]
.text:005411C5	push	esi
.text:005411C6	push	edi
.text:005411C7		
.text:005411C7 loop_begin:		

.text:005411C7	movsx	eax, byte ptr [ebp+0]
.text:005411CB	push	eax ; C
.text:005411CC	call	_tolower
.text:005411D1	add	esp, 4

```
.text:005411D4          cmp     al, 'a'
.text:005411D6          jl      short ↴
    ↳ next_character_in_password
.text:005411D8          cmp     al, 'z'
.text:005411DA          jg      short ↴
    ↳ next_character_in_password
.text:005411DC          movsx   ecx, al
```

```
.text:005411DF          sub     ecx, 'a' ; 97
```

```
.text:005411E2          cmp     ecx, 24
.text:005411E5          jle    short skip_subtracting
.text:005411E7          sub     ecx, 24
```

```
.text:005411EA
.text:005411EA skip_subtracting: ; CODE ↴
    ↳ XREF: rotate_all_with_password+35
```

```
.text:005411EA          mov     eax, 55555556h
.text:005411EF          imul   ecx
.text:005411F1          mov     eax, edx
.text:005411F3          shr    eax, 1Fh
.text:005411F6          add    edx, eax
.text:005411F8          mov     eax, ecx
.text:005411FA          mov     esi, edx
.text:005411FC          mov     ecx, 3
.text:00541201          cdq
.text:00541202          idiv   ecx
```

```
.text:00541204 sub     edx, 0
.text:00541207 jz      short call_rotate1 ;
.text:00541209 dec    edx
.text:0054120A jz      short call_rotate2 ; ..
.text:0054120C dec    edx
.text:0054120D jnz    short next_character_in_password
.text:0054120F test   ebx, ebx
.text:00541211 jle    short next_character_in_password
.text:00541213 mov    edi, ebx
```

```
.text:00541215 call_rotate3:
.text:00541215          push    esi
.text:00541216          call    rotate3
.text:0054121B          add    esp, 4
```

```

.text:0054121E          dec    edi
.text:0054121F          jnz    short call_rotate3
.text:00541221          jmp    short ↵
    ↳ next_character_in_password
.text:00541223
.text:00541223 call_rotate2:
.text:00541223           test   ebx, ebx
.text:00541225           jle    short ↵
    ↳ next_character_in_password
.text:00541227           mov    edi, ebx
.text:00541229
.text:00541229 loc_541229:
.text:00541229           push   esi
.text:0054122A           call   rotate2
.text:0054122F           add    esp, 4
.text:00541232           dec    edi
.text:00541233           jnz    short loc_541229
.text:00541235           jmp    short ↵
    ↳ next_character_in_password
.text:00541237
.text:00541237 call_rotate1:
.text:00541237           test   ebx, ebx
.text:00541239           jle    short ↵
    ↳ next_character_in_password
.text:0054123B           mov    edi, ebx
.text:0054123D
.text:0054123D loc_54123D:
.text:0054123D           push   esi
.text:0054123E           call   rotate1
.text:00541243           add    esp, 4
.text:00541246           dec    edi
.text:00541247           jnz    short loc_54123D
.text:00541249

```

```

.text:00541249 next_character_in_password:
.text:00541249           mov    al, [ebp+1]

```

```

.text:0054124C           inc    ebp
.text:0054124D           test   al, al
.text:0054124F           jnz    loop_begin
.text:00541255           pop    edi
.text:00541256           pop    esi
.text:00541257           pop    ebx
.text:00541258
.text:00541258 exit:
.text:00541258           pop    ebp
.text:00541259           retn

```

```
.text:00541259 rotate_all_with_password endp
```

```
void rotate_all (char *pwd, int v)
{
    char *p=pwd;

    while (*p)
    {
        char c=*p;
        int q;

        c=tolower (c);

        if (c>='a' && c<='z')
        {
            q=c-'a';
            if (q>24)
                q-=24;

            int quotient=q/3;
            int remainder=q % 3;

            switch (remainder)
            {
                case 0: for (int i=0; i<v; i++) rotate1
                ↵ (quotient); break;
                case 1: for (int i=0; i<v; i++) rotate2
                ↵ (quotient); break;
                case 2: for (int i=0; i<v; i++) rotate3
                ↵ (quotient); break;
            };
        };

        p++;
    };
}
```

.text:00541050 get_bit	proc near
.text:00541050	= dword ptr 4
.text:00541050 arg_0	= dword ptr 8
.text:00541050 arg_4	= byte ptr 0Ch
.text:00541050	
.text:00541050	mov eax, [esp+arg_4]
.text:00541054	mov ecx, [esp+arg_0]
.text:00541058	mov al, cube64[eax+ecx*8]
.text:0054105F	mov cl, [esp+arg_8]

.text:00541063	shr	al, cl
.text:00541065	and	al, 1
.text:00541067	retn	
.text:00541067 get_bit	endp	

:*arg_4 + arg_0 * 8.*

, set_bit():

.text:00541000 set_bit	proc	near
.text:00541000		
.text:00541000 arg_0	= dword ptr	4
.text:00541000 arg_4	= dword ptr	8
.text:00541000 arg_8	= dword ptr	0Ch
.text:00541000 arg_C	= byte ptr	10h
.text:00541000	mov	al, [esp+arg_C]
.text:00541004	mov	ecx, [esp+arg_8]
.text:00541008	push	esi
.text:00541009	mov	esi, [esp+4+arg_0]
.text:0054100D	test	al, al
.text:0054100F	mov	eax, [esp+4+arg_4]
.text:00541013	mov	dl, 1
.text:00541015	jz	short loc_54102B

.text:00541017	shl	dl, cl
.text:00541019	mov	cl, cube64[eax+esi*8]

.text:00541020	or	cl, dl
----------------	----	--------

.text:00541022	mov	cube64[eax+esi*8], cl
.text:00541029	pop	esi
.text:0054102A	retn	
.text:0054102B		
.text:0054102B loc_54102B:		
.text:0054102B	shl	dl, cl

.text:0054102D	mov	cl, cube64[eax+esi*8]
----------------	-----	-----------------------

.text:00541034	not	dl
----------------	-----	----

.text:00541036	and	cl, dl
----------------	-----	--------

```
.text:00541038          mov     cube64[eax+esi*8], cl
.text:0054103F          pop    esi
.text:00541040          retn
.text:00541040  set_bit    endp
```

```
#define IS_SET(flag, bit)      ((flag) & (bit))
#define SET_BIT(var, bit)       ((var) |= (bit))
#define REMOVE_BIT(var, bit)    ((var) &= ~(bit))

static BYTE cube[8][8];

void set_bit (int x, int y, int shift, int bit)
{
    if (bit)
        SET_BIT (cube[x][y], 1<<shift);
    else
        REMOVE_BIT (cube[x][y], 1<<shift);
};

bool get_bit (int x, int y, int shift)
{
    if ((cube[x][y]>>shift)&1==1)
        return 1;
    return 0;
};
```

```
.text:00541070 rotate1      proc near
.text:00541070
```

```
.text:00541070 internal_array_64= byte ptr -40h
.text:00541070 arg_0         = dword ptr 4
.text:00541070
.text:00541070          sub     esp, 40h
.text:00541073          push    ebx
.text:00541074          push    ebp
.text:00541075          mov     ebp, [esp+48h+arg_0]
.text:00541079          push    esi
.text:0054107A          push    edi
.text:0054107B          xor     edi, edi           ; EDI is ↴
    ↴ loop1 counter
```

EBX

```
.text:0054107D          lea     ebx, [esp+50h+ ↴
    ↴ internal_array_64]
.text:00541081
```

```
.text:00541081 first_loop1_begin:
.text:00541081     xor      esi, esi          ;
.text:00541083
.text:00541083 first_loop2_begin:
.text:00541083     push     ebp             ; arg_0
.text:00541084     push     esi             ;
.text:00541085     push     edi             ;
.text:00541086     call     get_bit
.text:0054108B     add      esp, 0Ch
.text:0054108E     mov      [ebx+esi], al
.text:00541091     inc      esi             ;
.text:00541092     cmp      esi, 8
.text:00541095     jl      short first_loop2_begin
.text:00541097     inc      edi             ;

;
.text:00541098     add      ebx, 8
.text:0054109B     cmp      edi, 8
.text:0054109E     jl      short first_loop1_begin
```

```
.text:005410A0     lea      ebx, [esp+50h+internal_array_64]
.text:005410A4     mov      edi, 7           ;
.text:005410A9
.text:005410A9 second_loop1_begin:
.text:005410A9     xor      esi, esi          ;
.text:005410AB
.text:005410AB second_loop2_begin:
.text:005410AB     mov      al, [ebx+esi]    ; EN(value from ↴
     ↴ internal array)
.text:005410AE     push     eax
.text:005410AF     push     ebp             ; arg_0
.text:005410B0     push     edi             ;
.text:005410B1     push     esi             ;
.text:005410B2     call     set_bit
.text:005410B7     add      esp, 10h
.text:005410BA     inc      esi             ;
.text:005410BB     cmp      esi, 8
.text:005410BE     jl      short second_loop2_begin
.text:005410C0     dec      edi             ;
.text:005410C1     add      ebx, 8           ;
.text:005410C4     cmp      edi, 0FFFFFFFh
.text:005410C7     jg      short second_loop1_begin
.text:005410C9     pop      edi
.text:005410CA     pop      esi
.text:005410CB     pop      ebp
```

```
.text:005410CC    pop     ebx
.text:005410CD    add     esp, 40h
.text:005410D0    retn
.text:005410D0    rotate1      endp
```

```
void rotate1 (int v)
{
    bool tmp[8][8]; // internal array
    int i, j;

    for (i=0; i<8; i++)
        for (j=0; j<8; j++)
            tmp[i][j]=get_bit (i, j, v);

    for (i=0; i<8; i++)
        for (j=0; j<8; j++)
            set_bit (j, 7-i, v, tmp[x][y]);
}
```

```
.text:005410E0 rotate2 proc near
.text:005410E0
.text:005410E0 internal_array_64 = byte ptr -40h
.text:005410E0 arg_0 = dword ptr 4
.text:005410E0
.text:005410E0    sub     esp, 40h
.text:005410E3    push    ebx
.text:005410E4    push    ebp
.text:005410E5    mov     ebp, [esp+48h+arg_0]
.text:005410E9    push    esi
.text:005410EA    push    edi
.text:005410EB    xor     edi, edi      ;
.text:005410ED    lea     ebx, [esp+50h+internal_array_64]
.text:005410F1
.text:005410F1 loc_5410F1:
.text:005410F1    xor     esi, esi      ;
.text:005410F3
.text:005410F3 loc_5410F3:
.text:005410F3    push    esi      ;
.text:005410F4    push    edi      ;
.text:005410F5    push    ebp      ; arg_0
.text:005410F6    call    get_bit
.text:005410FB    add     esp, 0Ch
.text:005410FE    mov     [ebx+esi], al      ;
.text:00541101    inc     esi      ;
.text:00541102    cmp     esi, 8
.text:00541105    jl     short loc_5410F3
```

```
.text:00541107    inc     edi          ;  
.text:00541108    add     ebx, 8  
.text:0054110B    cmp     edi, 8  
.text:0054110E    jl      short loc_5410F1  
.text:00541110    lea     ebx, [esp+50h+internal_array_64]  
.text:00541114    mov     edi, 7          ;  
.text:00541119  
.text:00541119 loc_541119:  
.text:00541119    xor     esi, esi       ;  
.text:0054111B  
.text:0054111B loc_54111B:  
.text:0054111B    mov     al, [ebx+esi]    ;  
.text:0054111E    push    eax  
.text:0054111F    push    edi          ;  
.text:00541120    push    esi          ;  
.text:00541121    push    ebp          ; arg_0  
.text:00541122    call    set_bit  
.text:00541127    add     esp, 10h  
.text:0054112A    inc     esi          ;  
.text:0054112B    cmp     esi, 8  
.text:0054112E    jl      short loc_54111B  
.text:00541130    dec     edi          ;  
.text:00541131    add     ebx, 8  
.text:00541134    cmp     edi, 0FFFFFFFh  
.text:00541137    jg      short loc_541119  
.text:00541139    pop     edi  
.text:0054113A    pop     esi  
.text:0054113B    pop     ebp  
.text:0054113C    pop     ebx  
.text:0054113D    add     esp, 40h  
.text:00541140    retn  
.text:00541140 rotate2 endp
```

```
void rotate2 (int v)  
{  
    bool tmp[8][8]; // internal array  
    int i, j;  
  
    for (i=0; i<8; i++)  
        for (j=0; j<8; j++)  
            tmp[i][j]=get_bit (v, i, j);  
  
    for (i=0; i<8; i++)  
        for (j=0; j<8; j++)  
            set_bit (v, j, 7-i, tmp[i][j]);  
};
```

```
void rotate3 (int v)
{
    bool tmp[8][8];
    int i, j;

    for (i=0; i<8; i++)
        for (j=0; j<8; j++)
            tmp[i][j]=get_bit (i, v, j);

    for (i=0; i<8; i++)
        for (j=0; j<8; j++)
            set_bit (7-j, v, i, tmp[i][j]);
}
```

¹?

```
void decrypt (BYTE *buf, int sz, char *pw)
{
    char *p=strdup (pw);
    strrev (p);
    int i=0;

    do
    {
        memcpy (cube, buf+i, 8*8);
        rotate_all (p, 3);
        memcpy (buf+i, cube, 8*8);
        i+=64;
    }
    while (i<sz);

    free (p);
}
```

strrev() ²

```
q=c-'a';
if (q>24)
    q-=24;

int quotient=q/3; // in range 0..7
int remainder=q % 3;

switch (remainder)
```

¹wikipedia²MSDN

```
{
    case 0: for (int i=0; i<v; i++) rotate1 (quotient); break; \
    ↴ // front
    case 1: for (int i=0; i<v; i++) rotate2 (quotient); break; \
    ↴ // top
    case 2: for (int i=0; i<v; i++) rotate3 (quotient); break; \
    ↴ // left
};
```

```
#include <windows.h>
#include <stdio.h>
#include <assert.h>

#define IS_SET(flag, bit)      ((flag) & (bit))
#define SET_BIT(var, bit)      ((var) |= (bit))
#define REMOVE_BIT(var, bit)   ((var) &= ~(bit))

static BYTE cube[8][8];

void set_bit (int x, int y, int z, bool bit)
{
    if (bit)
        SET_BIT (cube[x][y], 1<<z);
    else
        REMOVE_BIT (cube[x][y], 1<<z);
};

bool get_bit (int x, int y, int z)
{
    if ((cube[x][y]>>z)&1==1)
        return true;
    return false;
};

void rotate_f (int row)
{
    bool tmp[8][8];
    int x, y;

    for (x=0; x<8; x++)
        for (y=0; y<8; y++)
            tmp[x][y]=get_bit (x, y, row);

    for (x=0; x<8; x++)
        for (y=0; y<8; y++)
            set_bit (y, 7-x, row, tmp[x][y]);
};
```

```
};

void rotate_t (int row)
{
    bool tmp[8][8];
    int y, z;

    for (y=0; y<8; y++)
        for (z=0; z<8; z++)
            tmp[y][z]=get_bit (row, y, z);

    for (y=0; y<8; y++)
        for (z=0; z<8; z++)
            set_bit (row, z, 7-y, tmp[y][z]);
};

void rotate_l (int row)
{
    bool tmp[8][8];
    int x, z;

    for (x=0; x<8; x++)
        for (z=0; z<8; z++)
            tmp[x][z]=get_bit (x, row, z);

    for (x=0; x<8; x++)
        for (z=0; z<8; z++)
            set_bit (7-z, row, x, tmp[x][z]);
};

void rotate_all (char *pwd, int v)
{
    char *p=pwd;

    while (*p)
    {
        char c=*p;
        int q;

        c=tolower (c);

        if (c>='a' && c<='z')
        {
            q=c-'a';
            if (q>24)
                q-=24;
        }
    }
}
```

```

        int quotient=q/3;
        int remainder=q % 3;

        switch (remainder)
        {
            case 0: for (int i=0; i<v; i++) ↵
↳ rotate_f (quotient); break;
            case 1: for (int i=0; i<v; i++) ↵
↳ rotate_t (quotient); break;
            case 2: for (int i=0; i<v; i++) ↵
↳ rotate_l (quotient); break;
        };
    };

    p++;
};

void crypt (BYTE *buf, int sz, char *pw)
{
    int i=0;

    do
    {
        memcpy (cube, buf+i, 8*8);
        rotate_all (pw, 1);
        memcpy (buf+i, cube, 8*8);
        i+=64;
    }
    while (i<sz);
};

void decrypt (BYTE *buf, int sz, char *pw)
{
    char *p=strdup (pw);
    strrev (p);
    int i=0;

    do
    {
        memcpy (cube, buf+i, 8*8);
        rotate_all (p, 3);
        memcpy (buf+i, cube, 8*8);
        i+=64;
    }
    while (i<sz);
};

```

```
        free (p);
};

void crypt_file(char *fin, char* fout, char *pw)
{
    FILE *f;
    int flen, flen_aligned;
    BYTE *buf;

    f=fopen(fin, "rb");

    if (f==NULL)
    {
        printf ("Cannot open input file!\n");
        return;
    };

    fseek (f, 0, SEEK_END);
    flen=f.tell (f);
    fseek (f, 0, SEEK_SET);

    flen_aligned=(flen&0xFFFFFC0)+0x40;

    buf=(BYTE*)malloc (flen_aligned);
    memset (buf, 0, flen_aligned);

    fread (buf, flen, 1, f);

    fclose (f);

    crypt (buf, flen_aligned, pw);

    f=fopen(fout, "wb");

    fwrite ("QR9", 3, 1, f);
    fwrite (&flen, 4, 1, f);
    fwrite (buf, flen_aligned, 1, f);

    fclose (f);

    free (buf);
};

void decrypt_file(char *fin, char* fout, char *pw)
{
    FILE *f;
```

```
int real_flen,flen;
BYTE *buf;

f=fopen(fin, "rb");

if (f==NULL)
{
    printf ("Cannot open input file!\n");
    return;
};

fseek (f, 0, SEEK_END);
flen=f.tell (f);
fseek (f, 0, SEEK_SET);

buf=(BYTE*)malloc (flen);

fread (buf,flen,1,f);

fclose (f);

if (memcmp (buf, "QR9", 3)!=0)
{
    printf ("File is not encrypted!\n");
    return;
};

memcpy (&real_flen, buf+3, 4);

decrypt (buf+(3+4), flen-(3+4), pw);

f=fopen(fout, "wb");

fwrite (buf+(3+4), real_flen, 1, f);

fclose (f);

free (buf);
};

// run: input output 0/1 password
// 0 for encrypt, 1 for decrypt

int main(int argc, char *argv[])
{
    if (argc!=5)
    {
```

```
        printf ("Incorrect parameters!\n");
        return 1;
    };

    if (strcmp (argv[3], "0")==0)
        crypt_file (argv[1], argv[2], argv[4]);
    else
        if (strcmp (argv[3], "1")==0)
            decrypt_file (argv[1], argv[2], argv[4]);
        else
            printf ("Wrong param %s\n", argv[3]);
    }

    return 0;
};
```

Capítulo 80

SAP

80.1

(¹ y ²).

:

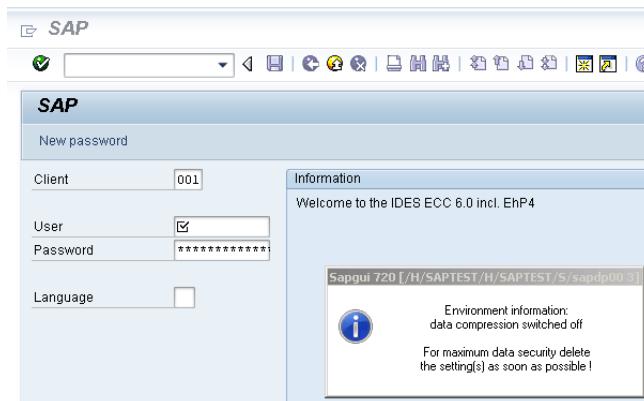


Figura 80.1:

¹<http://go.yurichev.com/17221>

²blog.yurichev.com

```
:
.text:6440D51B           lea    eax, [ebp+2108h+var_211C]
    ↴ ]
.text:6440D51E           push   eax          ; int
.text:6440D51F           push   offset aTdw_nocompress ; ↴
    ↴ "TDW_NOCOMPRESS"
.text:6440D524           mov    byte ptr [edi+15h], 0
.text:6440D528           call   chk_env
.text:6440D52D           pop    ecx
.text:6440D52E           pop    ecx
.text:6440D52F           push   offset byte_64443AF8
.text:6440D534           lea    ecx, [ebp+2108h+var_211C]
    ↴ ]

; demangled name: int ATL::CStringT::Compare(char const *)const
.text:6440D537           call   ds:mfc90_1603
.text:6440D53D           test   eax, eax
.text:6440D53F           jz    short loc_6440D55A
.text:6440D541           lea    ecx, [ebp+2108h+var_211C]
    ↴ ]

; demangled name: const char* ATL::CSimpleStringT::operator <
    ↴ PCXSTR
.text:6440D544           call   ds:mfc90_910
.text:6440D54A           push   eax          ; Str
.text:6440D54B           call   ds:atoi
.text:6440D551           test   eax, eax
.text:6440D553           setnz  al
.text:6440D556           pop    ecx
.text:6440D557           mov    [edi+15h], al
```

```
:
.text:64413F20 ; int __cdecl chk_env(char *VarName, int)
.text:64413F20 chk_env         proc near
.text:64413F20
.text:64413F20 DstSize        = dword ptr -0Ch
.text:64413F20 var_8          = dword ptr -8
.text:64413F20 DstBuf         = dword ptr -4
.text:64413F20 VarName        = dword ptr 8
.text:64413F20 arg_4          = dword ptr 0Ch
.text:64413F20
.text:64413F20             push   ebp
.text:64413F21             mov    ebp, esp
.text:64413F23             sub    esp, 0Ch
```

```

.text:64413F26          mov     [ebp+DstSize], 0
.text:64413F2D          mov     [ebp+DstBuf], 0
.text:64413F34          push    offset unk_6444C88C
.text:64413F39          mov     ecx, [ebp+arg_4]

; (demangled name) ATL::CStringT::operator=(char const *)
.text:64413F3C          call    ds:mfc90_820
.text:64413F42          mov     eax, [ebp+VarName]
.text:64413F45          push    eax           ; ↴
    ↴ VarName
.text:64413F46          mov     ecx, [ebp+DstSize]
.text:64413F49          push    ecx           ; ↴
    ↴ DstSize
.text:64413F4A          mov     edx, [ebp+DstBuf]
.text:64413F4D          push    edx           ; DstBuf
.text:64413F4E          lea     eax, [ebp+DstSize]
.text:64413F51          push    eax           ; ↴
    ↴ ReturnSize
.text:64413F52          call    ds:getenv_s
.text:64413F58          add    esp, 10h
.text:64413F5B          mov     [ebp+var_8], eax
.text:64413F5E          cmp     [ebp+var_8], 0
.text:64413F62          jz    short loc_64413F68
.text:64413F64          xor    eax, eax
.text:64413F66          jmp    short loc_64413FBC
.text:64413F68
.text:64413F68 loc_64413F68:
.text:64413F68          cmp     [ebp+DstSize], 0
.text:64413F6C          jnz    short loc_64413F72
.text:64413F6E          xor    eax, eax
.text:64413F70          jmp    short loc_64413FBC
.text:64413F72
.text:64413F72 loc_64413F72:
.text:64413F72          mov     ecx, [ebp+DstSize]
.text:64413F75          push   ecx
.text:64413F76          mov     ecx, [ebp+arg_4]

; demangled name: ATL::CSimpleStringT<char, 1>::Preallocate(int)
    ↴
.text:64413F79          call    ds:mfc90_2691
.text:64413F7F          mov     [ebp+DstBuf], eax
.text:64413F82          mov     edx, [ebp+VarName]
.text:64413F85          push    edx           ; ↴
    ↴ VarName
.text:64413F86          mov     eax, [ebp+DstSize]
.text:64413F89          push    eax           ; ↴
    ↴ DstSize

```

```

.text:64413F8A          mov    ecx, [ebp+DstBuf]
.text:64413F8D          push   ecx      ; DstBuf
.text:64413F8E          lea    edx, [ebp+DstSize]
.text:64413F91          push   edx      ; ↵
    ↳ ReturnSize
.text:64413F92          call   ds:getenv_s
.text:64413F98          add    esp, 10h
.text:64413F9B          mov    [ebp+var_8], eax
.text:64413F9E          push   0xFFFFFFFFh
.text:64413FA0          mov    ecx, [ebp+arg_4]

; demangled name: ATL::CSimpleStringT::ReleaseBuffer(int)
.text:64413FA3          call   ds:mfc90_5835
.text:64413FA9          cmp    [ebp+var_8], 0
.text:64413FAD          jz    short loc_64413FB3
.text:64413FAF          xor    eax, eax
.text:64413FB1          jmp   short loc_64413FBC
.text:64413FB3
.text:64413FB3 loc_64413FB3:
.text:64413FB3          mov    ecx, [ebp+arg_4]

; demangled name: const char* ATL::CSimpleStringT::operator ↵
    ↳ PCXSTR
.text:64413FB6          call   ds:mfc90_910
.text:64413FBC
.text:64413FBC loc_64413FBC:
.text:64413FBC
.text:64413FBC          mov    esp, ebp
.text:64413FBE          pop    ebp
.text:64413FBF          retn
.text:64413FBF chk_env  endp

```

. getenv_s()³ .

.

:

³MSDN

DPTRACE	“GUI-OPTION: Trace set to %d”
TDW_HEXDUMP	“GUI-OPTION: Hexdump enabled”
TDW_WORKDIR	“GUI-OPTION: working directory ‘%s’”
TDW_SPLASHSCREENOFF	“GUI-OPTION: Splash Screen Off” / “GUI-OPTION: Splash Screen Off”
TDW_REPLYTIMEOUT	“GUI-OPTION: reply timeout %d milliseconds”
TDW_PLAYBACKTIMEOUT	“GUI-OPTION: PlaybackTimeout set to %d milliseconds”
TDW_NOCOMPRESS	“GUI-OPTION: no compression read”
TDW_EXPERT	“GUI-OPTION: expert mode”
TDW_PLAYBACKPROGRESS	“GUI-OPTION: PlaybackProgress”
TDW_PLAYBACKNETTRAFFIC	“GUI-OPTION: PlaybackNetTraffic”
TDW_PLAYLOG	“GUI-OPTION: /PlayLog is YES, file %s”
TDW_PLAYTIME	“GUI-OPTION: /PlayTime set to %d milliseconds”
TDW_LOGFILE	“GUI-OPTION: TDW_LOGFILE ‘%s’”
TDW_WAN	“GUI-OPTION: WAN - low speed connection enabled”
TDW_FULLSCREEN	“GUI-OPTION: FullMenu enabled”
SAP_CP / SAP_CODEPAGE	“GUI-OPTION: SAP_CODEPAGE ‘%d’”
UPDOWNLOAD_CP	“GUI-OPTION: UPDOWNLOAD_CP ‘%d’”
SNC_PARTNERNAME	“GUI-OPTION: SNC name ‘%s’”
SNC_QOP	“GUI-OPTION: SNC_QOP ‘%s’”
SNC_LIB	“GUI-OPTION: SNC is set to: %s”
SAPGUI_INPLACE	“GUI-OPTION: environment variable SAPGUI_INPLACE is on”

:

```
.text:6440EE00          lea     edi, [ebp+2884h+var_2884]
    ↳ ] ; options here like +0x15...
.text:6440EE03          lea     ecx, [esi+24h]
.text:6440EE06          call    load_command_line
.text:6440EE0B          mov     edi, eax
.text:6440EE0D          xor     ebx, ebx
.text:6440EE0F          cmp     edi, ebx
.text:6440EE11          jz    short loc_6440EE42
.text:6440EE13          push    edi
.text:6440EE14          push    offset aSapguiStoppedA ; ↳
    ↳ "Sapgui stopped after commandline interp"...
.text:6440EE19          push    dword_644F93E8
.text:6440EE1F          call    FEWTraceError
```

Sí, y la única referencia está en
CDwsGui::PrepareInfoWindow():

```
.text:64405160          push    dword ptr [esi+2854h]
.text:64405166          push    offset aCdwsguiPrepare ; ↳
    ↳ "\nCDwsGui::PrepareInfoWindow: sapgui env"...
.text:6440516B          push    dword ptr [esi+2848h]
.text:64405171          call    dbg
.text:64405176          add     esp, 0Ch
```

...0:

```
.text:6440237A          push    eax
.text:6440237B          push    offset aCclientStart_6 ; \
    ↳ "CClient::Start: set shortcut user to '\%..." 
.text:64402380          push    dword ptr [edi+4]
.text:64402383          call    dbg
.text:64402388          add     esp, 0Ch
```

.

:

```
.text:64404F4F CDwsGui__PrepareInfoWindow proc near
.text:64404F4F
.text:64404F4F pvParam      = byte ptr -3Ch
.text:64404F4F var_38       = dword ptr -38h
.text:64404F4F var_34       = dword ptr -34h
.text:64404F4F rc           = tagRECT ptr -2Ch
.text:64404F4F cy           = dword ptr -1Ch
.text:64404F4F h            = dword ptr -18h
.text:64404F4F var_14       = dword ptr -14h
.text:64404F4F var_10       = dword ptr -10h
.text:64404F4F var_4        = dword ptr -4
.text:64404F4F
.text:64404F4F             push    30h
.text:64404F51             mov     eax, offset loc_64438E00
.text:64404F56             call    __EH_prolog3
.text:64404F5B             mov     esi, ecx      ; ECX is \
    ↳ pointer to object
.text:64404F5D             xor    ebx, ebx
.text:64404F5F             lea    ecx, [ebp+var_14]
.text:64404F62             mov    [ebp+var_10], ebx

; demangled name: ATL::CStringT(void)
.text:64404F65             call    ds:mfc90_316
.text:64404F6B             mov    [ebp+var_4], ebx
.text:64404F6E             lea    edi, [esi+2854h]
.text:64404F74             push   offset aEnvironmentInf ; \
    ↳ "Environment information:\n"
.text:64404F79             mov    ecx, edi

; demangled name: ATL::CStringT::operator=(char const *)
.text:64404F7B             call    ds:mfc90_820
.text:64404F81             cmp    [esi+38h], ebx
.text:64404F84             mov    ebx, ds:mfc90_2539
```

```

.text:64404F8A          jbe    short loc_64404FA9
.text:64404F8C          push   dword ptr [esi+34h]
.text:64404F8F          lea    eax, [ebp+var_14]
.text:64404F92          push   offset aWorkingDirecto ; \
    ↳ "working directory: '\%s'\n"
.text:64404F97          push   eax

; demangled name: ATL::CStringT::Format(char const *,...)
.text:64404F98          call   ebx ; mfc90_2539
.text:64404F9A          add    esp, 0Ch
.text:64404F9D          lea    eax, [ebp+var_14]
.text:64404FA0          push   eax
.text:64404FA1          mov    ecx, edi

; demangled name: ATL::CStringT::operator+=(class ATL::\
    ↳ CSimpleStringT<char, 1> const &)
.text:64404FA3          call   ds:mfc90_941
.text:64404FA9
.text:64404FA9 loc_64404FA9:
.text:64404FA9          mov    eax, [esi+38h]
.text:64404FAC          test   eax, eax
.text:64404FAE          jbe    short loc_64404FD3
.text:64404FB0          push   eax
.text:64404FB1          lea    eax, [ebp+var_14]
.text:64404FB4          push   offset aTraceLevelDAct ; \
    ↳ "trace level \%d activated\n"
.text:64404FB9          push   eax

; demangled name: ATL::CStringT::Format(char const *,...)
.text:64404FBA          call   ebx ; mfc90_2539
.text:64404FBC          add    esp, 0Ch
.text:64404FBF          lea    eax, [ebp+var_14]
.text:64404FC2          push   eax
.text:64404FC3          mov    ecx, edi

; demangled name: ATL::CStringT::operator+=(class ATL::\
    ↳ CSimpleStringT<char, 1> const &)
.text:64404FC5          call   ds:mfc90_941
.text:64404FCB          xor    ebx, ebx
.text:64404FCD          inc    ebx
.text:64404FCE          mov    [ebp+var_10], ebx
.text:64404FD1          jmp    short loc_64404FD6
.text:64404FD3
.text:64404FD3 loc_64404FD3:
.text:64404FD3          xor    ebx, ebx
.text:64404FD5          inc    ebx
.text:64404FD6

```

```

.text:64404FD6 loc_64404FD6:
.text:64404FD6 cmp [esi+38h], ebx
.text:64404FD9 jbe short loc_64404FF1
.text:64404FDB cmp dword ptr [esi+2978h], 0
.text:64404FE2 jz short loc_64404FF1
.text:64404FE4 push offset aHexdumpInTrace ; \
    ↴ "hexdump in trace activated\n"
.text:64404FE9 mov ecx, edi

; demangled name: ATL::CStringT::operator+=(char const *)
.text:64404FEB call ds:mfc90_945
.text:64404FF1
.text:64404FF1 loc_64404FF1:
.text:64404FF1 cmp byte ptr [esi+78h], 0
.text:64404FF5 jz short loc_64405007
.text:64404FF7 push offset aLoggingActivat ; \
    ↴ "logging activated\n"
.text:64404FFC mov ecx, edi

; demangled name: ATL::CStringT::operator+=(char const *)
.text:64404FFE call ds:mfc90_945
.text:64405004 mov [ebp+var_10], ebx
.text:64405007
.text:64405007 loc_64405007:
.text:64405007 cmp byte ptr [esi+3Dh], 0
.text:6440500B jz short bypass
.text:6440500D push offset aDataCompressio ; \
    ↴ "data compression switched off\n"
.text:64405012 mov ecx, edi

; demangled name: ATL::CStringT::operator+=(char const *)
.text:64405014 call ds:mfc90_945
.text:6440501A mov [ebp+var_10], ebx
.text:6440501D
.text:6440501D bypass:
.text:6440501D mov eax, [esi+20h]
.text:64405020 test eax, eax
.text:64405022 jz short loc_6440503A
.text:64405024 cmp dword ptr [eax+28h], 0
.text:64405028 jz short loc_6440503A
.text:6440502A push offset aDataRecordMode ; \
    ↴ "data record mode switched on\n"
.text:6440502F mov ecx, edi

; demangled name: ATL::CStringT::operator+=(char const *)
.text:64405031 call ds:mfc90_945

```

```

.text:64405037          mov     [ebp+var_10], ebx
.text:6440503A
.text:6440503A loc_6440503A:
.text:6440503A
.text:6440503A          mov     ecx, edi
.text:6440503C          cmp     [ebp+var_10], ebx
.text:6440503F          jnz    loc_64405142
.text:64405045          push    offset aForMaximumData ; \
    ↴ " \nFor maximum data security delete\nthe s"...

; demangled name: ATL::CStringT::operator+=(char const *)
.text:6440504A          call    ds:mfc90_945
.text:64405050          xor    edi, edi
.text:64405052          push    edi           ; \
    ↴ fWinIni
.text:64405053          lea    eax, [ebp+pvParam]
.text:64405056          push    eax           ; \
    ↴ pvParam
.text:64405057          push    edi           ; \
    ↴ uiParam
.text:64405058          push    30h           ; \
    ↴ uiAction
.text:6440505A          call    ds:SystemParametersInfoA
.text:64405060          mov    eax, [ebp+var_34]
.text:64405063          cmp    eax, 1600
.text:64405068          jle    short loc_64405072
.text:6440506A          cdq
.text:6440506B          sub    eax, edx
.text:6440506D          sar    eax, 1
.text:6440506F          mov    [ebp+var_34], eax
.text:64405072
.text:64405072 loc_64405072:
.text:64405072          push    edi           ; hWnd
.text:64405073          mov    [ebp+cy], 0A0h
.text:6440507A          call    ds:GetDC
.text:64405080          mov    [ebp+var_10], eax
.text:64405083          mov    ebx, 12Ch
.text:64405088          cmp    eax, edi
.text:6440508A          jz    loc_64405113
.text:64405090          push    11h           ; i
.text:64405092          call    ds:GetStockObject
.text:64405098          mov    edi, ds:SelectObject
.text:6440509E          push    eax           ; h
.text:6440509F          push    [ebp+var_10]      ; hdc
.text:644050A2          call    edi ; SelectObject
.text:644050A4          and    [ebp+rc.left], 0
.text:644050A8          and    [ebp+rc.top], 0

```

```

.text:644050AC          mov    [ebp+h], eax
.text:644050AF          push   401h           ; format
.text:644050B4          lea    eax, [ebp+rc]
.text:644050B7          push   eax           ; lprc
.text:644050B8          lea    ecx, [esi+2854h]
.text:644050BE          mov    [ebp+rc.right], ebx
.text:644050C1          mov    [ebp+rc.bottom], 0B4h

; demangled name: ATL::CSimpleStringT::GetLength(void)
.text:644050C8          call   ds:mfc90_3178
.text:644050CE          push   eax           ; ↵
    ↳ cchText
.text:644050CF          lea    ecx, [esi+2854h]

; demangled name: const char* ATL::CSimpleStringT::operator ↵
    ↳ PCXSTR
.text:644050D5          call   ds:mfc90_910
.text:644050DB          push   eax           ; ↵
    ↳ lpchText
.text:644050DC          push   [ebp+var_10] ; hdc
.text:644050DF          call   ds:DrawTextA
.text:644050E5          push   4             ; nIndex
.text:644050E7          call   ds:GetSystemMetrics
.text:644050ED          mov    ecx, [ebp+rc.bottom]
.text:644050F0          sub    ecx, [ebp+rc.top]
.text:644050F3          cmp    [ebp+h], 0
.text:644050F7          lea    eax, [eax+ecx+28h]
.text:644050FB          mov    [ebp+cy], eax
.text:644050FE          jz    short loc_64405108
.text:64405100          push   [ebp+h]       ; h
.text:64405103          push   [ebp+var_10] ; hdc
.text:64405106          call   edi ; SelectObject
.text:64405108          loc_64405108:
.text:64405108          push   [ebp+var_10] ; hDC
.text:6440510B          push   0             ; hWnd
.text:6440510D          call   ds:ReleaseDC
.text:64405113          loc_64405113:
.text:64405113          mov    eax, [ebp+var_38]
.text:64405116          push   80h           ; uFlags
.text:6440511B          push   [ebp+cy]       ; cy
.text:6440511E          inc    eax
.text:6440511F          push   ebx           ; cx
.text:64405120          push   eax           ; Y
.text:64405121          mov    eax, [ebp+var_34]
.text:64405124          add    eax, 0FFFFFED4h

```

```

.text:64405129          cdq
.text:6440512A          sub    eax, edx
.text:6440512C          sar    eax, 1
.text:6440512E          push   eax           ; X
.text:6440512F          push   0             ; ↵
    ↳ hWndInsertAfter
.text:64405131          push   dword ptr [esi+285Ch] ; ↵
    ↳ hWnd
.text:64405137          push   ds:SetWindowPos
.text:6440513D          xor    ebx, ebx
.text:6440513F          inc    ebx
.text:64405140          jmp    short loc_6440514D
.text:64405142
.text:64405142 loc_64405142:
.text:64405142          push   offset byte_64443AF8

; demangled name: ATL::CStringT::operator=(char const *)
.text:64405147          call   ds:mfc90_820
.text:6440514D
.text:6440514D loc_6440514D:
.text:6440514D          cmp    dword_6450B970, ebx
.text:64405153          jl    short loc_64405188
.text:64405155          call   sub_6441C910
.text:6440515A          mov    dword_644F858C, ebx
.text:64405160          push   dword ptr [esi+2854h]
.text:64405166          push   offset aCdwsGuiPrepare ; ↵
    ↳ "\nCDwsGui::PrepareInfoWindow: sapgui env"...
.text:6440516B          push   dword ptr [esi+2848h]
.text:64405171          call   dbg
.text:64405176          add    esp, 0Ch
.text:64405179          mov    dword_644F858C, 2
.text:64405183          call   sub_6441C920
.text:64405188
.text:64405188 loc_64405188:
.text:64405188          or     [ebp+var_4], 0FFFFFFFh
.text:6440518C          lea    ecx, [ebp+var_14]

; demangled name: ATL::CStringT::~CStringT()
.text:6440518F          call   ds:mfc90_601
.text:64405195          call   __EH_epilog3
.text:6440519A          retn
.text:6440519A CDwsGui__PrepareInfoWindow endp

```

:

```

.text:64405007 loc_64405007:
.text:64405007          cmp    byte ptr [esi+3Dh], 0
.text:6440500B          jz    short bypass

```

```
.text:6440500D          push    offset aDataCompressio ; \
    ↴ "data compression switched off\n"
.text:64405012          mov     ecx, edi
; demangled name: ATL::CStringT::operator+=(char const *)
.text:64405014          call    ds:mfc90_945
.text:6440501A          mov     [ebp+var_10], ebx
.text:6440501D
.text:6440501D bypass:
```

:

```
.text:6440503C          cmp     [ebp+var_10], ebx
.text:6440503F          jnz    exit ;
; "For maximum data security delete" / "the setting(s) as soon \
    ↴ as possible !":
.text:64405045          push    offset aForMaximumData ; \
    ↴ "\nFor maximum data security delete\nthe s"...
.text:6440504A          call    ds:mfc90_945 ; ATL:: \
    ↴ CStringT::operator+=(char const *)
.text:64405050          xor    edi, edi
.text:64405052          push    edi           ; \
    ↴ fWinIni
.text:64405053          lea     eax, [ebp+pvParam]
.text:64405056          push    eax           ; \
    ↴ pvParam
.text:64405057          push    edi           ; \
    ↴ uiParam
.text:64405058          push    30h           ; \
    ↴ uiAction
.text:6440505A          call    ds:SystemParametersInfoA
.text:64405060          mov     eax, [ebp+var_34]
.text:64405063          cmp     eax, 1600
.text:64405068          jle    short loc_64405072
.text:6440506A          cdq
.text:6440506B          sub    eax, edx
.text:6440506D          sar    eax, 1
.text:6440506F          mov     [ebp+var_34], eax
.text:64405072          loc_64405072:
; :
.text:64405072          push    edi           ; hWnd
.text:64405073          mov     [ebp+cy], 0A0h
```

.text:6440507A	call	ds:GetDC
----------------	------	----------

JNZ ...

.text:6440503F	jnz	exit ;
----------------	-----	--------

...

.text:64404C19 sub_64404C19	proc	near
.text:64404C19		= dword ptr 4
.text:64404C19 arg_0		
.text:64404C19	push	ebx
.text:64404C1A	push	ebp
.text:64404C1B	push	esi
.text:64404C1C	push	edi
.text:64404C1D	mov	edi, [esp+10h+arg_0]
.text:64404C21	mov	eax, [edi]
.text:64404C23	mov	esi, ecx ; ESI/ECX are ↴ ↳ pointers to some unknown object.
.text:64404C25	mov	[esi], eax
.text:64404C27	mov	eax, [edi+4]
.text:64404C2A	mov	[esi+4], eax
.text:64404C2D	mov	eax, [edi+8]
.text:64404C30	mov	[esi+8], eax
.text:64404C33	lea	eax, [edi+0Ch]
.text:64404C36	push	eax
.text:64404C37	lea	ecx, [esi+0Ch]
; demangled name: ATL::CStringT::operator=(class ATL::CStringT ↴ ↳ ... &)		
.text:64404C3A	call	ds:mfc90_817
.text:64404C40	mov	eax, [edi+10h]
.text:64404C43	mov	[esi+10h], eax
.text:64404C46	mov	al, [edi+14h]
.text:64404C49	mov	[esi+14h], al
.text:64404C4C	mov	al, [edi+15h] ; copy ↴ ↳ byte from 0x15 offset
.text:64404C4F	mov	[esi+15h], al ; to 0x15 ↴ ↳ offset in CDwsGui object

CDwsGui::Init():

.text:6440B0BF loc_6440B0BF:		
.text:6440B0BF	mov	eax, [ebp+arg_0]

```
.text:6440B0C2          push    [ebp+arg_4]
.text:6440B0C5          mov     [esi+2844h], eax
.text:6440B0CB          lea     eax, [esi+28h] ; ESI is ↴
    ↳ pointer to CDwsGui object
.text:6440B0CE          push    eax
.text:6440B0CF          call    CDwsGui__CopyOptions
```

```
.text:64409D58          cmp     [esi+3Dh], bl   ; ESI is ↴
    ↳ pointer to CDwsGui object
.text:64409D5B          lea     ecx, [esi+2B8h]
.text:64409D61          setz    al
.text:64409D64          push    eax           ; arg_10 ↴
    ↳ of CConnectionContext::CreateNetwork
.text:64409D65          push    dword ptr [esi+64h]

; demangled name: const char* ATL::CSimpleStringT::operator ↴
    ↳ PCXSTR
.text:64409D68          call    ds:mfc90_910
.text:64409D68          ; no ↴
    ↳ arguments
.text:64409D6E          push    eax
.text:64409D6F          lea     ecx, [esi+2BCh]

; demangled name: const char* ATL::CSimpleStringT::operator ↴
    ↳ PCXSTR
.text:64409D75          call    ds:mfc90_910
.text:64409D75          ; no ↴
    ↳ arguments
.text:64409D7B          push    eax
.text:64409D7C          push    esi
.text:64409D7D          lea     ecx, [esi+8]
.text:64409D80          call    ↴
    ↳ CConnectionContext__CreateNetwork
```

```
...
.text:64403476          push    [ebp+compression]
.text:64403479          push    [ebp+arg_C]
.text:6440347C          push    [ebp+arg_8]
.text:6440347F          push    [ebp+arg_4]
.text:64403482          push    [ebp+arg_0]
.text:64403485          call    CNetwork__CNetwork
```

```
.text:64411DF1          cmp     [ebp+compression], esi
.text:64411DF7          jz      short set_EAX_to_0
.text:64411DF9          mov     al, [ebx+78h] ; ↴
    ↳ another value may affect compression?
```

```

.text:64411DFC          cmp    al, '3'
.text:64411DFE          jz     short set_EAX_to_1
.text:64411E00          cmp    al, '4'
.text:64411E02          jnz   short set_EAX_to_0
.text:64411E04
.text:64411E04 set_EAX_to_1:
.text:64411E04          xor    eax, eax
.text:64411E06          inc    eax           ; EAX -> ↴
    ↴ 1
.text:64411E07          jmp    short loc_64411E0B
.text:64411E09
.text:64411E09 set_EAX_to_0:
.text:64411E09          xor    eax, eax       ; EAX -> ↴
    ↴ 0
.text:64411E0B
.text:64411E0B loc_64411E0B:
.text:64411E0B          mov    [ebx+3A4h], eax ; EBX is ↴
    ↴ pointer to CNetwork object

```

```

.text:64406F76 loc_64406F76:
.text:64406F76          mov    ecx, [ebp+7728h+var_7794] ↴
    ↴ ]
.text:64406F79          cmp    dword ptr [ecx+3A4h], 1
.text:64406F80          jnz   compression_flag_is_zero
.text:64406F86          mov    byte ptr [ebx+7], 1
.text:64406F8A          mov    eax, [esi+18h]
.text:64406F8D          mov    ecx, eax
.text:64406F8F          test   eax, eax
.text:64406F91          ja    short loc_64406FFF
.text:64406F93          mov    ecx, [esi+14h]
.text:64406F96          mov    eax, [esi+20h]
.text:64406F99
.text:64406F99 loc_64406F99:
.text:64406F99          push   dword ptr [edi+2868h] ; ↴
    ↴ int
.text:64406F9F          lea    edx, [ebp+7728h+var_77A4] ↴
    ↴ ]
.text:64406FA2          push   edx           ; int
.text:64406FA3          push   30000          ; int
.text:64406FA8          lea    edx, [ebp+7728h+Dst]
.text:64406FAB          push   edx           ; Dst
.text:64406FAC          push   ecx           ; int
.text:64406FAD          push   eax           ; Src
.text:64406FAE          push   dword ptr [edi+28C0h] ; ↴
    ↴ int
.text:64406FB4          call   sub_644055C5      ; ↴

```

```

    ↳ actual compression routine
.text:64406FB9          add     esp, 1Ch
.text:64406FBC          cmp     eax, 0FFFFFFF6h
.text:64406FBF          jz    short loc_64407004
.text:64406FC1          cmp     eax, 1
.text:64406FC4          jz    loc_6440708C
.text:64406FCA          cmp     eax, 2
.text:64406FCD          jz    short loc_64407004
.text:64406FCF          push    eax
.text:64406FD0          push    offset aCompressionErr ; ↳
    ↳ "compression error [rc = %d]- program wi"...
.text:64406FD5          push    offset aGui_err_compre ; ↳
    ↳ "GUI_ERR_COMPRESS"
.text:64406FDA          push    dword ptr [edi+28D0h]
.text:64406FE0          call    SapPcTxtRead

```

```

.text:6441747C          push    offset aErrorCsSrcCompre ; ↳
    ↳ "\nERROR: CsRCompress: invalid handle"
.text:64417481          call    eax ; dword_644F94C8
.text:64417483          add     esp, 4

```

Voilà! ⁴,

.text:64406F79	cmp	dword ptr [ecx+3A4h], 1
.text:64406F80	jnz	compression_flag_is_zero

80.2

«Password logon no longer possible - too many failed attempts», .

TYPEINFODUMP⁵.

FUNCTION ThVmcsEvent				
Address:	10143190	Size:	675 bytes	Index:
↳ 60483	TypeIndex:	60484		
Type:	int NEAR_C ThVmcsEvent	(unsigned int, unsigned char,		
↳	unsigned short*)			
Flags:	0			
PARAMETER events				
Address:	Reg335+288	Size:	4 bytes	Index:
↳ TypeIndex:	60489			
Type:	unsigned int			
Flags:	d0			

⁴<http://go.yurichev.com/17312>

⁵<http://go.yurichev.com/17038>

PARAMETER opcode
 Address: Reg335+296 Size: 1 bytes Index: 60490 ↵
 ↳ TypeIndex: 60491
 Type: unsigned char
Flags: d0

PARAMETER serverName
 Address: Reg335+304 Size: 8 bytes Index: 60492 ↵
 ↳ TypeIndex: 60493
 Type: unsigned short*

Flags: d0

STATIC_LOCAL_VAR func
 Address: 12274af0 Size: 8 bytes Index: ↵
 ↳ 60495 TypeIndex: 60496
 Type: wchar_t*

Flags: 80

LOCAL_VAR admhead
 Address: Reg335+304 Size: 8 bytes Index: 60498 ↵
 ↳ TypeIndex: 60499
 Type: unsigned char*

Flags: 90

LOCAL_VAR record
 Address: Reg335+64 Size: 204 bytes Index: 60501 ↵
 ↳ TypeIndex: 60502
 Type: AD_RECORD

Flags: 90

LOCAL_VAR adlen
 Address: Reg335+296 Size: 4 bytes Index: 60508 ↵
 ↳ TypeIndex: 60509
 Type: int

Flags: 90

STRUCT DBSL_STMTID
 Size: 120 Variables: 4 Functions: 0 Base classes: 0

MEMBER moduletype
 Type: DBSL_MODULETYPE
 Offset: 0 Index: 3 TypeIndex: 38653

MEMBER module
 Type: wchar_t module[40]
 Offset: 4 Index: 3 TypeIndex: 831

MEMBER stmtnum
 Type: long
 Offset: 84 Index: 3 TypeIndex: 440

MEMBER timestamp
 Type: wchar_t timestamp[15]
 Offset: 88 Index: 3 TypeIndex: 6612

6

..

:

```

cmp    cs:ct_level, 1
jl     short loc_1400375DA
call   DpLock
lea    rcx, aDpxxtool4_c ; "dpxxtool4.c"
mov    edx, 4Eh           ; line
call   CTrcSaveLocation
mov    r8, cs:func_48
mov    rcx, cs:hdl        ; hdl
lea    rdx, aSDpreadmemvalu ; "%s: DpReadMemValue (%d)"
mov    r9d, ebx
call   DpTrcErr
call   DpUnlock

```

```
cat "disp+work.pdb.d" | grep FUNCTION | grep -i password
```

```

FUNCTION rcui::AgiPassword::DiagISelection
FUNCTION ssf_password_encrypt
FUNCTION ssf_password_decrypt
FUNCTION password_logon_disabled
FUNCTION dySignSkipUserPassword
FUNCTION migrate_password_history
FUNCTION password_is_initial
FUNCTION rcui::AgiPassword::IsVisible
FUNCTION password_distance_ok
FUNCTION get_password_downwards_compatibility
FUNCTION dySignUnSkipUserPassword
FUNCTION rcui::AgiPassword::GetTypeName
FUNCTION `rcui::AgiPassword::AgiPassword'::`1'::dtor$2
FUNCTION `rcui::AgiPassword::AgiPassword'::`1'::dtor$0
FUNCTION `rcui::AgiPassword::AgiPassword'::`1'::dtor$1
FUNCTION usm_set_password
FUNCTION rcui::AgiPassword::TraceTo
FUNCTION days_since_last_password_change
FUNCTION rsecgrp_generate_random_password
FUNCTION rcui::AgiPassword::`scalar deleting destructor'
FUNCTION password_attempt_limit_exceeded
FUNCTION handle_incorrect_password
FUNCTION `rcui::AgiPassword::`scalar deleting destructor'
    ↴ ``::`1'::dtor$1
FUNCTION calculate_new_password_hash
FUNCTION shift_password_to_history
FUNCTION rcui::AgiPassword::GetType

```

⁶: <http://go.yurichev.com/17039>

```

FUNCTION found_password_in_history
FUNCTION `rcui::AgiPassword::`scalar deleting destructor ↵
    ↴ '::`1'::dtor$0
FUNCTION rcui::AgiObj::IsaPassword
FUNCTION password_idle_check
FUNCTION SlicHwPasswordForDay
FUNCTION rcui::AgiPassword::IsaPassword
FUNCTION rcui::AgiPassword::AgiPassword
FUNCTION delete_user_password
FUNCTION usm_set_user_password
FUNCTION Password_API
FUNCTION get_password_change_for_SSO
FUNCTION password_in_USR40
FUNCTION rsec_agrp_abap_generate_random_password

```

*«user was locked by subsequently failed password logon attempts»
password_attempt_limit_exceeded().*

: «*password logon attempt will be rejected immediately (preventing dictionary attacks)*»,
«failed-logon lock: expired (but not removed due to 'read-only' operation)», *«failed-logon lock: expired => removed»*.

..

:

tracer:

```

tracer64.exe -a:disp+work.exe bpf=disp+work.exe!chckpass,args ↵
    ↴ :3,unicode

```

```

PID=2236|TID=2248|(0) disp+work.exe!chckpass (0x202c770, L" ↵
    ↴ Brewered1                                     ", 0x41) (called ↵
    ↴ from 0x1402f1060 (disp+work.exe!usrexist+0x3c0))
PID=2236|TID=2248|(0) disp+work.exe!chckpass -> 0x35

```

:syssigni() -> DylSigni() -> dychkusr() -> usrexist() -> chckpass().

0x35 chckpass():

.text:00000001402ED567 loc_1402ED567:			↗
↴ ; CODE XREF: chckpass+B4			
.text:00000001402ED567	mov	rcx, rbx	↗
↴ ; usr02			
.text:00000001402ED56A	call		↗
↴ password_idle_check			
.text:00000001402ED56F	cmp	eax, 33h	
.text:00000001402ED572	jz	loc_1402EDB4E	

```
.text:00000001402ED578          cmp    eax, 36h
.text:00000001402ED57B          jz     loc_1402EDB3D
.text:00000001402ED581          xor    edx, edx
    ↳ ; usr02_READONLY
.text:00000001402ED583          mov    rcx, rbx
    ↳ ; usr02
.text:00000001402ED586          call   ↵
    ↳ password_attempt_limit_exceeded
.text:00000001402ED58B          test   al, al
.text:00000001402ED58D          jz     short ↵
    ↳ loc_1402ED5A0
.text:00000001402ED58F          mov    eax, 35h
.text:00000001402ED594          add    rsp, 60h
.text:00000001402ED598          pop    r14
.text:00000001402ED59A          pop    r12
.text:00000001402ED59C          pop    rdi
.text:00000001402ED59D          pop    rsi
.text:00000001402ED59E          pop    rbx
.text:00000001402ED59F          retn
```

```
: tracer64.exe -a:disp+work.exe bpf=disp+work.exe! ↴  
    ↴ password_attempt_limit_exceeded,args:4,unicode,rt:0
```

```
PID=2744|TID=360|(0) disp+work.exe! ↴
    ↳ password_attempt_limit_exceeded (0x202c770, 0, 0x257758, ↵
    ↳ 0) (called from 0x1402ed58b (disp+work.exe!chckpass+0xeb)) ↵
    ↳ )
PID=2744|TID=360|(0) disp+work.exe! ↴
    ↳ password_attempt_limit_exceeded -> 1
PID=2744|TID=360|We modify return value (EAX/RAX) of this ↵
    ↳ function to 0
PID=2744|TID=360|(0) disp+work.exe! ↴
    ↳ password_attempt_limit_exceeded (0x202c770, 0, 0, 0) ( ↵
    ↳ called from 0x1402e9794 (disp+work.exe!chngpass+0xe4)) ↵
PID=2744|TID=360|(0) disp+work.exe! ↴
    ↳ password_attempt_limit_exceeded -> 1
PID=2744|TID=360|We modify return value (EAX/RAX) of this ↵
    ↳ function to 0
```

```
tracer64.exe -a:disp+work.exe bpf=disp+work.exe!chckpass,args ↴ :3,unicode,rt:0
```

PID=2744|TID=360|(0) disp+work.exe!chckpass (0x202c770, L"bogus", 0x41) (called from 0x1402f1060 (disp+work.exe!usrexist+0x3c0))

```
PID=2744|TID=360|(0) disp+work.exe!chckpass -> 0x35  
PID=2744|TID=360|We modify return value (EAX/RAX) of this ↴  
↳ function to 0
```

```
lea      rcx, aLoginFailed_us ; "login/failed_user_auto_unlock"  
call    sapgparam  
test    rax, rax  
jz      short loc_1402E19DE  
movzx  eax, word ptr [rax]  
cmp    ax, 'N'  
jz      short loc_1402E19D4  
cmp    ax, 'n'  
jz      short loc_1402E19D4  
cmp    ax, '0'  
jnz    short loc_1402E19DE
```

Capítulo 81

Oracle RDBMS

81.1

```
SQL> select * from V$VERSION;
```

:

```
BANNER
```

```
-----  
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - ✓  
  ↳ Production  
PL/SQL Release 11.2.0.1.0 - Production  
CORE    11.2.0.1.0      Production  
TNS for 32-bit Windows: Version 11.2.0.1.0 - Production  
NLSRTL Version 11.2.0.1.0 - Production
```

V\$VERSION?

:

Listing 81.1: kqf.o

```
.rodata:0800C4A0 kqfviw          dd 0Bh           ; DATA✓  
  ↳ XREF: kqfchk:loc_8003A6D  
.rodata:0800C4A0                  ; ✓  
  ↳ kqfgbn+34  
.rodata:0800C4A4          dd offset _2__STRING_10102_0 ;✓  
  ↳ "GV$WAITSTAT"  
.rodata:0800C4A8          dd 4
```

```

.rodata:0800C4AC          dd offset _2__STRING_10103_0 ; \
    ↴ "NULL"
.rodata:0800C4B0          dd 3
.rodata:0800C4B4          dd 0
.rodata:0800C4B8          dd 195h
.rodata:0800C4BC          dd 4
.rodata:0800C4C0          dd 0
.rodata:0800C4C4          dd 0FFF1CBh
.rodata:0800C4C8          dd 3
.rodata:0800C4CC          dd 0
.rodata:0800C4D0          dd 0Ah
.rodata:0800C4D4          dd offset _2__STRING_10104_0 ; \
    ↴ "V$WAITSTAT"
.rodata:0800C4D8          dd 4
.rodata:0800C4DC          dd offset _2__STRING_10103_0 ; \
    ↴ "NULL"
.rodata:0800C4E0          dd 3
.rodata:0800C4E4          dd 0
.rodata:0800C4E8          dd 4Eh
.rodata:0800C4EC          dd 3
.rodata:0800C4F0          dd 0
.rodata:0800C4F4          dd 0FFF003h
.rodata:0800C4F8          dd 4
.rodata:0800C4FC          dd 0
.rodata:0800C500          dd 5
.rodata:0800C504          dd offset _2__STRING_10105_0 ; \
    ↴ "GV$BH"
.rodata:0800C508          dd 4
.rodata:0800C50C          dd offset _2__STRING_10103_0 ; \
    ↴ "NULL"
.rodata:0800C510          dd 3
.rodata:0800C514          dd 0
.rodata:0800C518          dd 269h
.rodata:0800C51C          dd 15h
.rodata:0800C520          dd 0
.rodata:0800C524          dd 0FFF1EDh
.rodata:0800C528          dd 8
.rodata:0800C52C          dd 0
.rodata:0800C530          dd 4
.rodata:0800C534          dd offset _2__STRING_10106_0 ; \
    ↴ "V$BH"
.rodata:0800C538          dd 4
.rodata:0800C53C          dd offset _2__STRING_10103_0 ; \
    ↴ "NULL"
.rodata:0800C540          dd 3
.rodata:0800C544          dd 0
.rodata:0800C548          dd 0F5h

```

.rodata:0800C54C	dd 14h
.rodata:0800C550	dd 0
.rodata:0800C554	dd 0FFFFC1EEh
.rodata:0800C558	dd 5
.rodata:0800C55C	dd 0

. . . : «6 significant initial characters in an external identifier»¹

V\$FIXED_VIEW_DEFINITION

```
SQL> select * from V$FIXED_VIEW_DEFINITION where view_name='
  ↴ V$VERSION';
```

VIEW_NAME

VIEW_DEFINITION

V\$VERSION

```
select BANNER from GV$VERSION where inst_id = USERENV('
  ↴ Instance')
```

```
SQL> select * from V$FIXED_VIEW_DEFINITION where view_name='
  ↴ GV$VERSION';
```

VIEW_NAME

VIEW_DEFINITION

GV\$VERSION

```
select inst_id, banner from x$version
```

```
select BANNER from GV$VERSION where inst_id = USERENV('Instance'
':
```

Listing 81.2: kqf.o

```
rodata:080185A0 kqfvip          dd offset _2__STRING_11126_0 ; 
  ↴ DATA XREF: kqfgvcn+18
.rodata:080185A0                  ; ;
  ↴ kqfgvt+F
.rodata:080185A0                  ; "
  ↴ select inst_id,decode(indx,1,'data bloc"...
```

¹Draft ANSI C Standard (ANSI X3J11/88-090) (May 13, 1988) (yurichev.com)

```
.rodata:080185A4          dd offset kqfv459_c_0
.rodata:080185A8          dd 0
.rodata:080185AC          dd 0

...
.rodata:08019570          dd offset _2__STRING_11378_0 ; \
    ↴ "select  BANNER from GV$VERSION where in"...
.rodata:08019574          dd offset kqfv133_c_0
.rodata:08019578          dd 0
.rodata:0801957C          dd 0
.rodata:08019580          dd offset _2__STRING_11379_0 ; \
    ↴ "select inst_id,decode(bitandcfflg,1),0"...
.rodata:08019584          dd offset kqfv403_c_0
.rodata:08019588          dd 0
.rodata:0801958C          dd 0
.rodata:08019590          dd offset _2__STRING_11380_0 ; \
    ↴ "select  STATUS , NAME, IS_RECOVERY_DEST"...
.rodata:08019594          dd offset kqfv199_c_0
```

Listing 81.3: kqf.o

```
.rodata:080BBAC4 kqfv133_c_0      dd 6                      ; DATA \
    ↴ XREF: .rodata:08019574
.rodata:080BBAC8              dd offset _2__STRING_5017_0 ; \
    ↴ "BANNER"
.rodata:080BBACC             dd 0
.rodata:080BBAD0              dd offset _2__STRING_0_0
```

Listing 81.4: oracle tables

```
kqfviw_element.viewname: [V$VERSION] ?: 0x3 0x43 0x1 0xfffffc085 \
    ↴ 0x4
kqfvip_element.statement: [select  BANNER from GV$VERSION where \
    ↴ inst_id = USERENV('Instance')]
kqfvip_element.params:
[BANNER]
```

y:

Listing 81.5: oracle tables

```
kqfviw_element.viewname: [GV$VERSION] ?: 0x3 0x26 0x2 0 \
    ↴ 0xfffffc192 0x1
kqfvip_element.statement: [select inst_id, banner from \
    ↴ x$version]
kqfvip_element.params:
[INST_ID] [BANNER]
```

```
SQL> select * from x$version;
          ADDR        INDX     INST_ID
----- -----
BANNER
----- 
          ↴
0DBAF574      0          1
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - ↴
          ↴ Production
...

```

Listing 81.6: kqf.o

```
.rodata:0803CAC0 dd 9 ; ↴ element number 0x1f6
.rodata:0803CAC4 dd offset _2__STRING_13113_0 ; ↴ "X$VERSION"
.rodata:0803CAC8 dd 4
.rodata:0803CACC dd offset _2__STRING_13114_0 ; ↴ "kqvt"
.rodata:0803CAD0 dd 4
.rodata:0803CAD4 dd 4
.rodata:0803CAD8 dd 0
.rodata:0803CADC dd 4
.rodata:0803CAE0 dd 0Ch
.rodata:0803CAE4 dd 0FFF075h
.rodata:0803CAE8 dd 3
.rodata:0803CAEC dd 0
.rodata:0803CAF0 dd 7
.rodata:0803CAF4 dd offset _2__STRING_13115_0 ; ↴ "X$KQFSZ"
.rodata:0803CAF8 dd 5
.rodata:0803CAF C dd offset _2__STRING_13116_0 ; ↴ "kqfsz"
.rodata:0803CB00 dd 1
.rodata:0803CB04 dd 38h
.rodata:0803CB08 dd 0
.rodata:0803CB0C dd 7
.rodata:0803CB10 dd 0
.rodata:0803CB14 dd 0FFF09Dh
.rodata:0803CB18 dd 2
.rodata:0803CB1C dd 0
```

kqf.o:

Listing 81.7: kqf.o

```
.rodata:0808C360 kqvt_c_0          kqftap_param <4, offset ↴
    ↳ _2_STRING_19_0, 917h, 0, 0, 0, 4, 0, 0>
.rodata:0808C360                                ; DATA ↴
    ↳ XREF: .rodata:08042680
.rodata:0808C360                                ; " ↴
    ↳ ADDR"
.rodata:0808C384          kqftap_param <4, offset ↴
    ↳ _2_STRING_20_0, 0B02h, 0, 0, 0, 4, 0, 0> ; "INDX"
.rodata:0808C3A8          kqftap_param <7, offset ↴
    ↳ _2_STRING_21_0, 0B02h, 0, 0, 0, 4, 0, 0> ; "INST_ID"
.rodata:0808C3CC          kqftap_param <6, offset ↴
    ↳ _2_STRING_5017_0, 601h, 0, 0, 0, 50h, 0, 0> ; "BANNER"
.rodata:0808C3F0          kqftap_param <0, offset ↴
    ↳ _2_STRING_0_0, 0, 0, 0, 0, 0, 0, 0>
```

Listing 81.8: kqf.o

```
.rodata:08042680          kqftap_element <0, offset ↴
    ↳ kqvt_c_0, offset kqvrow, 0> ; element 0x1f6
```

Listing 81.9: oracle tables

```
kqftab_element.name: [X$VERSION] ?: [kqvt] 0x4 0x4 0x4 0xc 0x
    ↳ xfffffc075 0x3
kqftap_param.name=[ADDR] ?: 0x917 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INDX] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INST_ID] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[BANNER] ?: 0x601 0x0 0x0 0x0 0x50 0x0 0x0
kqftap_element.fn1=kqvrow
kqftap_element.fn2=NULL
```

```
tracer -a:oracle.exe bpf=oracle.exe!_kqvrow,trace:cc
```

```
_kqvrow_ proc near

var_7C      = byte ptr -7Ch
var_18      = dword ptr -18h
var_14      = dword ptr -14h
Dest        = dword ptr -10h
var_C       = dword ptr -0Ch
var_8       = dword ptr -8
var_4       = dword ptr -4
arg_8       = dword ptr 10h
```

```

arg_C      = dword ptr  14h
arg_14     = dword ptr  1Ch
arg_18     = dword ptr  20h

; FUNCTION CHUNK AT .text1:056C11A0 SIZE 00000049 BYTES

    push    ebp
    mov     ebp, esp
    sub     esp, 7Ch
    mov     eax, [ebp+arg_14] ; [EBP+1Ch]=1
    mov     ecx, TlsIndex ; [69AEB08h]=0
    mov     edx, large fs:2Ch
    mov     edx, [edx+ecx*4] ; [EDX+ECX*4]=0xc98c938
    cmp     eax, 2          ; EAX=1
    mov     eax, [ebp+arg_8] ; [EBP+10h]=0xcdfe554
    jz    loc_2CE1288
    mov     ecx, [eax]       ; [EAX]=0..5
    mov     [ebp+var_4], edi ; EDI=0xc98c938

loc_2CE10F6: ; CODE XREF: _kqvrow_+10A
             ; _kqvrow_+1A9
    cmp     ecx, 5          ; ECX=0..5
    ja     loc_56C11C7
    mov     edi, [ebp+arg_18] ; [EBP+20h]=0
    mov     [ebp+var_14], edx ; EDX=0xc98c938
    mov     [ebp+var_8], ebx ; EBX=0
    mov     ebx, eax         ; EAX=0xcdfe554
    mov     [ebp+var_C], esi ; ESI=0xcdfe248

loc_2CE110D: ; CODE XREF: _kqvrow_+29E00E6
    mov     edx, ds:off_628B09C[ecx*4] ; [ECX*4+628B09Ch]
    ↴ ]=0x2ce1116, 0x2ce11ac, 0x2ce11db, 0x2ce11f6, 0x2ce1236, ↴
    ↴ 0x2ce127a
    jmp     edx           ; EDX=0x2ce1116, 0x2ce11ac, 0x2ce11db, 0x2ce11f6, 0x2ce1236, 0x2ce127a

loc_2CE1116: ; DATA XREF: .rdata:off_628B09C
    push    offset aXKqvvsnBuffer ; "$kqvvsn buffer"
    mov     ecx, [ebp+arg_C] ; [EBP+14h]=0x8a172b4
    xor     edx, edx
    mov     esi, [ebp+var_14] ; [EBP-14h]=0xc98c938
    push    edx           ; EDX=0
    push    edx           ; EDX=0
    push    50h
    push    ecx           ; ECX=0x8a172b4
    push    dword ptr [esi+10494h] ; [ESI+10494h]=0xc98cd58
    ↴ xc98cd58

```

```

    call    _kghalf          ; tracing nested maximum ↵
    ↳ level (1) reached, skipping this CALL
        mov     esi, ds:_imp_vsnum ; [59771A8h]=0x61bc49e0
        mov     [ebp+Dest], eax ; EAX=0xce2ffb0
        mov     [ebx+8], eax   ; EAX=0xce2ffb0
        mov     [ebx+4], eax   ; EAX=0xce2ffb0
        mov     edi, [esi]      ; [ESI]=0xb200100
        mov     esi, ds:_imp_vsnstr ; [597D6D4h]=0x65852148 ↵
    ↳ , "- Production"
        push    esi           ; ESI=0x65852148, "- ↵
    ↳ Production"
        mov     ebx, edi      ; EDI=0xb200100
        shr     ebx, 18h       ; EBX=0xb200100
        mov     ecx, edi      ; EDI=0xb200100
        shr     ecx, 14h       ; ECX=0xb200100
        and    ecx, 0Fh       ; ECX=0xb2
        mov     edx, edi      ; EDI=0xb200100
        shr     edx, 0Ch       ; EDX=0xb200100
        movzx  edx, dl       ; DL=0
        mov     eax, edi      ; EDI=0xb200100
        shr     eax, 8        ; EAX=0xb200100
        and    eax, 0Fh       ; EAX=0xb2001
        and    edi, 0FFh      ; EDI=0xb200100
        push    edi           ; EDI=0
        mov     edi, [ebp+arg_18] ; [EBP+20h]=0
        push    eax           ; EAX=1
        mov     eax, ds:_imp_vsnban ; [597D6D8h]=0x65852100 ↵
    ↳ , "Oracle Database 11g Enterprise Edition Release %d.%d.%d.%d.%d.%d.%s" ↵
    ↳ d.%d.%d %s"
        push    edx           ; EDX=0
        push    ecx           ; ECX=2
        push    ebx           ; EBX=0xb
        mov     ebx, [ebp+arg_8] ; [EBP+10h]=0xcdfe554
        push    eax           ; EAX=0x65852100, "Oracle ↵
    ↳ Database 11g Enterprise Edition Release %d.%d.%d.%d.%d.%d.%d %s" ↵
    ↳ "
        mov     eax, [ebp+Dest] ; [EBP-10h]=0xce2ffb0
        push    eax           ; EAX=0xce2ffb0
        call    ds:_imp_sprintf ; op1=MSVCR80.dll!sprintf ↵
    ↳ tracing nested maximum level (1) reached, skipping this ↵
    ↳ CALL
        add     esp, 38h
        mov     dword ptr [ebx], 1

loc_2CE1192: ; CODE XREF: _kqvrow_+FB
                ; _kqvrow_+128 ...
        test   edi, edi      ; EDI=0

```

```

jnz      __VIfreq_kqvrow
mov      esi, [ebp+var_C] ; [EBP-0Ch]=0xcdfe248
mov      edi, [ebp+var_4] ; [EBP-4]=0xc98c938
mov      eax, ebx          ; EBX=0xcdfe554
mov      ebx, [ebp+var_8] ; [EBP-8]=0
lea      eax, [eax+4]       ; [EAX+4]=0xce2ffb0, "NLSRTL "
                                ↴ Version 11.2.0.1.0 - Production", "Oracle Database 11g "
                                ↴ Enterprise Edition Release 11.2.0.1.0 - Production", "PL/
                                ↴ SQL Release 11.2.0.1.0 - Production", "TNS for 32-bit "
                                ↴ Windows: Version 11.2.0.1.0 - Production"

loc_2CE11A8: ; CODE XREF: _kqvrow_+29E00F6
    mov      esp, ebp
    pop      ebp
    retn             ; EAX=0xcdfe558

loc_2CE11AC: ; DATA XREF: .rdata:0628B0A0
    mov      edx, [ebx+8]     ; [EBX+8]=0xce2ffb0, "Oracle "
                                ↴ Database 11g Enterprise Edition Release 11.2.0.1.0 - 
                                ↴ Production"
        mov      dword ptr [ebx], 2
        mov      [ebx+4], edx      ; EDX=0xce2ffb0, "Oracle "
                                ↴ Database 11g Enterprise Edition Release 11.2.0.1.0 - 
                                ↴ Production"
        push     edx              ; EDX=0xce2ffb0, "Oracle "
                                ↴ Database 11g Enterprise Edition Release 11.2.0.1.0 - 
                                ↴ Production"
        call     _kkxvsn           ; tracing nested maximum "
                                ↴ level (1) reached, skipping this CALL
        pop      ecx
        mov      edx, [ebx+4]     ; [EBX+4]=0xce2ffb0, "PL/SQL "
                                ↴ Release 11.2.0.1.0 - Production"
        movzx   ecx, byte ptr [edx] ; [EDX]=0x50
        test    ecx, ecx          ; ECX=0x50
        jnz    short loc_2CE1192
        mov      edx, [ebp+var_14]
        mov      esi, [ebp+var_C]
        mov      eax, ebx
        mov      ebx, [ebp+var_8]
        mov      ecx, [eax]
        jmp     loc_2CE10F6

loc_2CE11DB: ; DATA XREF: .rdata:0628B0A4
    push     0
    push     50h
    mov      edx, [ebx+8]     ; [EBX+8]=0xce2ffb0, "PL/SQL "
                                ↴ Release 11.2.0.1.0 - Production"

```

```

        mov      [ebx+4], edx      ; EDX=0xce2ffb0, "PL/SQL ↵
↳ Release 11.2.0.1.0 - Production"
        push     edx              ; EDX=0xce2ffb0, "PL/SQL ↵
↳ Release 11.2.0.1.0 - Production"
        call     _lmxver          ; tracing nested maximum ↵
↳ level (1) reached, skipping this CALL
        add     esp, 0Ch
        mov     dword ptr [ebx], 3
        jmp     short loc_2CE1192

loc_2CE11F6: ; DATA XREF: .rdata:0628B0A8
        mov     edx, [ebx+8]      ; [EBX+8]=0xce2ffb0
        mov     [ebp+var_18], 50h
        mov     [ebx+4], edx      ; EDX=0xce2ffb0
        push    0
        call     _npinli          ; tracing nested maximum ↵
↳ level (1) reached, skipping this CALL
        pop     ecx
        test    eax, eax         ; EAX=0
        jnz    loc_56C11DA
        mov     ecx, [ebp+var_14] ; [EBP-14h]=0xc98c938
        lea     edx, [ebp+var_18] ; [EBP-18h]=0x50
        push    edx              ; EDX=0xd76c93c
        push    dword ptr [ebx+8] ; [EBX+8]=0xce2ffb0
        push    dword ptr [ecx+13278h]; [ECX+13278h]=0↙
↳ xacce190
        call     _nrttnsvrs       ; tracing nested maximum ↵
↳ level (1) reached, skipping this CALL
        add     esp, 0Ch

loc_2CE122B: ; CODE XREF: _kqvrow_+29E0118
        mov     dword ptr [ebx], 4
        jmp     loc_2CE1192

loc_2CE1236: ; DATA XREF: .rdata:0628B0AC
        lea     edx, [ebp+var_7C] ; [EBP-7Ch]=1
        push    edx              ; EDX=0xd76c8d8
        push    0
        mov     esi, [ebx+8]      ; [EBX+8]=0xce2ffb0, "TNS for ↵
↳ 32-bit Windows: Version 11.2.0.1.0 - Production"
        mov     [ebx+4], esi      ; ESI=0xce2ffb0, "TNS for 32- ↵
↳ bit Windows: Version 11.2.0.1.0 - Production"
        mov     ecx, 50h
        mov     [ebp+var_18], ecx ; ECX=0x50
        push    ecx              ; ECX=0x50
        push    esi              ; ESI=0xce2ffb0, "TNS for 32- ↵
↳ bit Windows: Version 11.2.0.1.0 - Production"

```

```

    call    _lxvers          ; tracing nested maximum ↵
↳ level (1) reached, skipping this CALL
    add    esp, 10h
    mov    edx, [ebp+var_18] ; [EBP-18h]=0x50
    mov    dword ptr [ebx], 5
    test   edx, edx         ; EDX=0x50
    jnz    loc_2CE1192
    mov    edx, [ebp+var_14]
    mov    esi, [ebp+var_C]
    mov    eax, ebx
    mov    ebx, [ebp+var_8]
    mov    ecx, 5
    jmp    loc_2CE10F6

loc_2CE127A: ; DATA XREF: .rdata:0628B0B0
    mov    edx, [ebp+var_14] ; [EBP-14h]=0xc98c938
    mov    esi, [ebp+var_C] ; [EBP-0Ch]=0xcdfe248
    mov    edi, [ebp+var_4] ; [EBP-4]=0xc98c938
    mov    eax, ebx         ; EBX=0xcdfe554
    mov    ebx, [ebp+var_8] ; [EBP-8]=0

loc_2CE1288: ; CODE XREF: _kqvrow_+1F
    mov    eax, [eax+8]      ; [EAX+8]=0xce2ffb0, "NLSRTL" ↵
↳ Version 11.2.0.1.0 - Production"
    test   eax, eax         ; EAX=0xce2ffb0, "NLSRTL" ↵
↳ Version 11.2.0.1.0 - Production"
    jz    short loc_2CE12A7
    push   offset aXKqvvsnBuffer ; "x$kqvvsn buffer"
    push   eax               ; EAX=0xce2ffb0, "NLSRTL" ↵
↳ Version 11.2.0.1.0 - Production"
    mov    eax, [ebp+arg_C] ; [EBP+14h]=0x8a172b4
    push   eax               ; EAX=0x8a172b4
    push   dword ptr [edx+10494h] ; [EDX+10494h]=0
↳ xc98cd58
    call   _kghfrf          ; tracing nested maximum ↵
↳ level (1) reached, skipping this CALL
    add    esp, 10h

loc_2CE12A7: ; CODE XREF: _kqvrow_+1C1
    xor    eax, eax
    mov    esp, ebp
    pop    ebp
    retn             ; EAX=0
_kqvrow_ endp

```

1	kkxvsn().
2	lmxver().
3	npinli(), nrtnsvrs().
4	lxvers().
5	

81.2 Oracle RDBMS

Diagnosing and Resolving Error ORA-04031 on the Shared Pool or Other Memory Pools [Video] [ID 146599.1] :

There is a fixed table called X\$KSMLRU that tracks allocations in the shared pool that cause other objects in the shared pool to be aged out. This fixed table can be used to identify what is causing the large allocation.

If many objects are being periodically flushed from the shared pool then this will cause response time problems and will likely cause library cache latch contention problems when the objects are reloaded into the shared pool.

One unusual thing about the X\$KSMLRU fixed table is that the contents of the fixed table are erased whenever someone selects from the fixed table. This is done since the fixed table stores only the largest allocations that have occurred. The values are reset after being selected so that subsequent large allocations can be noted even if they were not quite as large as others that occurred previously. Because of this resetting, the output of selecting from this table should be carefully kept since it cannot be retrieved back after the query is issued.

Listing 81.10: oracle tables

```
kqftab_element.name: [X$KSMLRU] ?: [ksmlr] 0x4 0x64 0x11 0xc 0x
↳ xffffc0bb 0x5
kqftap_param.name=[ADDR] ?: 0x917 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INDX] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INST_ID] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[KSMLRIDX] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[KSMLRDUR] ?: 0xb02 0x0 0x0 0x0 0x4 0x4 0x0
kqftap_param.name=[KSMLRSHRPOOL] ?: 0xb02 0x0 0x0 0x0 0x4 0x8 0x2
↳ x0
kqftap_param.name=[KSMLRCOM] ?: 0x501 0x0 0x0 0x0 0x14 0xc 0x0
kqftap_param.name=[KSMLRSIZ] ?: 0x2 0x0 0x0 0x0 0x4 0x20 0x0
kqftap_param.name=[KSMLRNUM] ?: 0x2 0x0 0x0 0x0 0x4 0x24 0x0
kqftap_param.name=[KSMLRHON] ?: 0x501 0x0 0x0 0x0 0x20 0x28 0x0
```

```
kqftap_param.name=[KSMLROHV] ?: 0xb02 0x0 0x0 0x0 0x4 0x48 0x0
kqftap_param.name=[KSMLRSES] ?: 0x17 0x0 0x0 0x0 0x4 0x4c 0x0
kqftap_param.name=[KSMLRADU] ?: 0x2 0x0 0x0 0x0 0x4 0x50 0x0
kqftap_param.name=[KSMLRNID] ?: 0x2 0x0 0x0 0x0 0x4 0x54 0x0
kqftap_param.name=[KSMLRNSD] ?: 0x2 0x0 0x0 0x0 0x4 0x58 0x0
kqftap_param.name=[KSMLRNCD] ?: 0x2 0x0 0x0 0x0 0x4 0x5c 0x0
kqftap_param.name=[KSMLRNED] ?: 0x2 0x0 0x0 0x0 0x4 0x60 0x0
kqftap_element.fn1=ksmlrs
kqftap_element.fn2=NULL
```

Listing 81.11: ksm.o

```
...
.text:00434C50 loc_434C50: ; DATA 
    ↳ XREF: .rdata:off_5E50EA8
.text:00434C50          mov     edx, [ebp-4]
.text:00434C53          mov     [eax], esi
.text:00434C55          mov     esi, [edi]
.text:00434C57          mov     [eax+4], esi
.text:00434C5A          mov     [edi], eax
.text:00434C5C          add     edx, 1
.text:00434C5F          mov     [ebp-4], edx
.text:00434C62          jnz    loc_434B7D
.text:00434C68          mov     ecx, [ebp+14h]
.text:00434C6B          mov     ebx, [ebp-10h]
.text:00434C6E          mov     esi, [ebp-0Ch]
.text:00434C71          mov     edi, [ebp-8]
.text:00434C74          lea     eax, [ecx+8Ch]
.text:00434C7A          push    370h      ; Size
.text:00434C7F          push    0          ; Val
.text:00434C81          push    eax        ; Dst
.text:00434C82          call    __intel_fast_memset
.text:00434C87          add     esp, 0Ch
.text:00434C8A          mov     esp, ebp
.text:00434C8C          pop     ebp
.text:00434C8D          retn
.text:00434C8D _ksmsplu endp
```

0x434C7A 0x434C8A.

```
tracer -a:oracle.exe bpx=oracle.exe!0x00434C7A, set(eip,0)
    ↳ x00434C8A)
```

81.3 Oracle RDBMS

V\$TIMER

V\$TIMER displays the elapsed time in hundredths of a second. Time is measured since the beginning of the epoch, which is operating system specific, and wraps around to 0 again whenever the value overflows four bytes (roughly 497 days).

(²)

```
SQL> select * from V$FIXED_VIEW_DEFINITION where view_name='
   ↴ V$TIMER';

VIEW_NAME
-----
VIEW_DEFINITION
-----
   ↴

V$TIMER
select HSECS from GV$TIMER where inst_id = USERENV('Instance')

SQL> select * from V$FIXED_VIEW_DEFINITION where view_name='
   ↴ GV$TIMER';

VIEW_NAME
-----
VIEW_DEFINITION
-----
   ↴

GV$TIMER
select inst_id,ksutmtim from x$ksutm
```

:

Listing 81.12: oracle tables

kqftab_element.name: [X\$KSUTM] ?: [ksutm] 0x1 0x4 0x4 0x0 0x ↴ xffffc09b 0x3
kqftap_param.name=[ADDR] ?: 0x10917 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INDX] ?: 0x20b02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[INST_ID] ?: 0xb02 0x0 0x0 0x0 0x4 0x0 0x0
kqftap_param.name=[KSUTMTIM] ?: 0x1302 0x0 0x0 0x0 0x4 0x0 0x1e

²<http://go.yurichev.com/17088>

```
kqftap_element.fn1=NULL
kqftap_element.fn2=NULL
```

```
kqfd_DRN_ksutm_c proc near ; DATA XREF: .rodata✓
    ↳ :0805B4E8

arg_0          = dword ptr  8
arg_8          = dword ptr  10h
arg_C          = dword ptr  14h

        push    ebp
        mov     ebp, esp
        push    [ebp+arg_C]
        push    offset ksugtm
        push    offset _2__STRING_1263_0 ; "KSUTMTIM"
        push    [ebp+arg_8]
        push    [ebp+arg_0]
        call    kqfd_cfui_drain
        add     esp, 14h
        mov     esp, ebp
        pop     ebp
        retn

kqfd_DRN_ksutm_c endp
```

kqfd_DRN_ksutm_c() kqfd_tab_registry_0 :

```
dd offset _2__STRING_62_0 ; "X$KSUTM"
dd offset kqfd_OPN_ksutm_c
dd offset kqfd_tabl_fetch
dd 0
dd 0
dd offset kqfd_DRN_ksutm_c
```

..

Listing 81.13: ksuo

```
ksugtm      proc near

var_1C       = byte ptr -1Ch
arg_4        = dword ptr  0Ch

        push    ebp
        mov     ebp, esp
        sub     esp, 1Ch
        lea     eax, [ebp+var_1C]
        push    eax
```

```

call    slgcs
pop    ecx
mov    edx, [ebp+arg_4]
mov    [edx], eax
mov    eax, 4
mov    esp, ebp
pop    ebp
retn
ksugtm    endp

```

```

tracer -a:oracle.exe bpf=oracle.exe!_ksugtm,args:2,dump_args:0 ↵
↳ x4

```

```
SQL> select * from V$TIMER;
```

```
HSECS
```

```
-----
```

```
27294929
```

```
SQL> select * from V$TIMER;
```

```
HSECS
```

```
-----
```

```
27295006
```

```
SQL> select * from V$TIMER;
```

```
HSECS
```

```
-----
```

```
27295167
```

Listing 81.14:

```

TID=2428|(0) oracle.exe!_ksugtm (0x0, 0xd76c5f0) (called from ↵
↳ oracle.exe!__VInfreq_qerfxFetch+0xfad (0x56bb6d5))
Argument 2/2
0D76C5F0: 38 C9          "8.    ↵
                           "
TID=2428|(0) oracle.exe!_ksugtm () -> 0x4 (0x4)
Argument 2/2 difference
00000000: D1 7C A0 01      ".|... ↵
                           "

```

```

TID=2428|(0) oracle.exe!_ksugtm (0x0, 0xd76c5f0) (called from ↵
    ↳ oracle.exe!__VInfreq_qerfxFetch+0xfad (0x56bb6d5))
Argument 2/2
0D76C5F0: 38 C9                                "8. ↵
    ↳           "
TID=2428|(0) oracle.exe!_ksugtm () -> 0x4 (0x4)
Argument 2/2 difference
00000000: 1E 7D A0 01                            ".}... ↵
    ↳           "
TID=2428|(0) oracle.exe!_ksugtm (0x0, 0xd76c5f0) (called from ↵
    ↳ oracle.exe!__VInfreq_qerfxFetch+0xfad (0x56bb6d5))
Argument 2/2
0D76C5F0: 38 C9                                "8. ↵
    ↳           "
TID=2428|(0) oracle.exe!_ksugtm () -> 0x4 (0x4)
Argument 2/2 difference
00000000: BF 7D A0 01                            ".}... ↵
    ↳           "

```

.

slgcs() (Linux x86):

```

slgcs          proc near

var_4          = dword ptr -4
arg_0          = dword ptr  8

        push    ebp
        mov     ebp, esp
        push    esi
        mov     [ebp+var_4], ebx
        mov     eax, [ebp+arg_0]
        call    $+5
        pop    ebx
        nop             ; PIC mode
        mov     ebx, offset _GLOBAL_OFFSET_TABLE_
        mov     dword ptr [eax], 0
        call    sltrgatime64    ; PIC mode
        push    0
        push    0Ah
        push    edx
        push    eax
        call    __udivdi3      ; PIC mode
        mov     ebx, [ebp+var_4]
        add     esp, 10h
        mov     esp, ebp
        pop    ebp

```

_slgcs	ret endp
--------	-------------

(`sltrgatime64()` 10 ([41 on page 631](#)))

```
:
_slgcs      proc near           ; CODE XREF: ↵
    ↳ _dbgefgHtElResetCount+15          ; _dbgerRunActions+1528
        db      66h
        nop
        push    ebp
        mov     ebp, esp
        mov     eax, [ebp+8]
        mov     dword ptr [eax], 0
        call    ds:_imp__GetTickCount@0 ; GetTickCount ↵
    ↳ ()                                ; _dbgerRunActions+1528
        mov     edx, eax
        mov     eax, 0CCCCCCCCDh
        mul     edx
        shr     edx, 3
        mov     eax, edx
        mov     esp, ebp
        pop     ebp
        retn
_slgcs      endp
```

`GetTickCount()` ³ 10 ([41 on page 631](#)).

`kqfd_tab_registry_0` oracle tables⁴, :

[X\$KSUTM] [kqfd_OPN_ksutm_c] [kqfd_tabl_fetch] [NULL] [NULL] [↗ ↳ kqfd_DRN_ksutm_c] [X\$KSUSGIF] [kqfd_OPN_ksusg_c] [kqfd_tabl_fetch] [NULL] [NULL] ↗ ↳ [kqfd_DRN_ksusg_c]

OPN, , open, DRN, drain.

³[MSDN](#)

⁴[yurichev.com](#)

Capítulo 82

82.1 EICAR

: «EICAR-STANDARD-ANTIVIRUS-TEST-FILE!»¹.

:

```
X50!P%@AP[4\PZX54(P^)7CC)7}$_EICAR-STANDARD-ANTIVIRUS-TEST-FILE!✓
↳ $H+H*
```

:

```
; : SP=0FFEh, SS:[SP]=0
0100 58          pop     ax
; AX=0, SP=0
0101 35 4F 21    xor     ax, 214Fh
; AX = 214Fh and SP = 0
0104 50          push    ax
; AX = 214Fh, SP = FFFEh and SS:[FFFE] = 214Fh
0105 25 40 41    and     ax, 4140h
; AX = 140h, SP = FFFEh and SS:[FFFE] = 214Fh
0108 50          push    ax
; AX = 140h, SP = FFFCh, SS:[FFFC] = 140h and SS:[FFFE] = 214Fh
0109 5B          pop     bx
; AX = 140h, BX = 140h, SP = FFFEh and SS:[FFFE] = 214Fh
010A 34 5C          xor     al, 5Ch
; AX = 11Ch, BX = 140h, SP = FFFEh and SS:[FFFE] = 214Fh
010C 50          push    ax
010D 5A          pop     dx
; AX = 11Ch, BX = 140h, DX = 11Ch, SP = FFFEh and SS:[FFFE] = ✓
↳ 214Fh
```

¹wikipedia

```

010E 58          pop     ax
; AX = 214Fh, BX = 140h, DX = 11Ch and SP = 0
010F 35 34 28    xor     ax, 2834h
; AX = 97Bh, BX = 140h, DX = 11Ch and SP = 0
0112 50          push    ax
0113 5E          pop     si
; AX = 97Bh, BX = 140h, DX = 11Ch, SI = 97Bh and SP = 0
0114 29 37        sub    [bx], si
0116 43          inc     bx
0117 43          inc     bx
0118 29 37        sub    [bx], si
011A 7D 24        jge    short near ptr word_10140
011C 45 49 43 ... db 'EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$'
0140 48 2B    word_10140 dw 2B48h ; CD 21 (INT 21)
0142 48 2A        dw 2A48h ; CD 20 (INT 20)
0144 0D          db 0Dh
0145 0A          db 0Ah

```

```

B4 09      MOV AH, 9
BA 1C 01    MOV DX, 11Ch
CD 21      INT 21h
CD 20      INT 20h

```

INT 21h DS:DX.. [CP/M](#).INT 20h DOS.

..

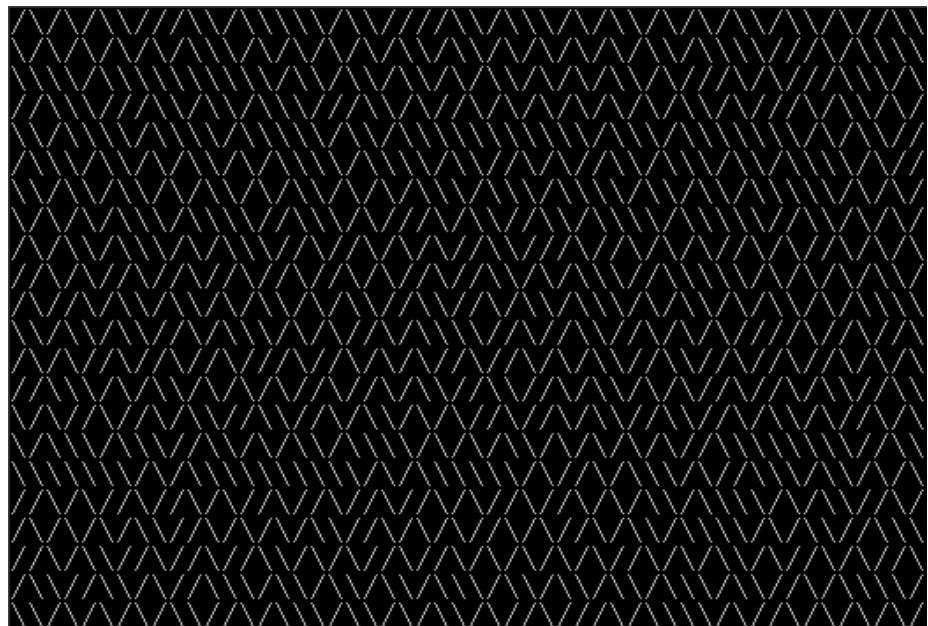
- ;
- ;
- INT 21 y INT 20.

: [A.6.5 on page 1244](#).

Capítulo 83

83.1 10 PRINT CHR\$(205.5+RND(1)); : GOTO 10

[a12] . :



83.1.1¹, .

```

00000000: B001      mov      al,1      ;
00000002: CD10      int      010      ;
00000004: 30FF      xor      bh,bh     ;
00000006: B9D007    mov      cx,007D0   ;
00000009: 31C0      xor      ax,ax     ;
0000000B: 9C        pushf    ;         ;
;
0000000C: FA        cli      ;         ;
0000000D: E643      out     043,al    ;
;
0000000F: E440      in       al,040    ;
00000011: 88C4      mov      ah,al     ;
00000013: E440      in       al,040    ;
00000015: 9D        popf    ;         ;
00000016: 86C4      xchg    ah,al     ;
;
00000018: D1E8      shr      ax,1     ;
0000001A: D1E8      shr      ax,1     ;
;
0000001C: B05C      mov      al,05C ;'\' ;
;
0000001E: 7202      jc      000000022  ;
;
00000020: B02F      mov      al,02F ;'/' ;
;
00000022: B40E      mov      ah,00E    ;
00000024: CD10      int      010      ;
00000026: E2E1      loop    000000009  ;
00000028: CD20      int      020      ;
;
```

43h, « 0», "counter latch", "" () .

POPF.

AL, .

83.1.2

. .

¹<http://go.yurichev.com/17305>

:

```

00000000: B9D007    mov     cx,007D0  ;
00000003: 31C0      xor     ax,ax   ;
00000005: E643      out    043,al  ;
00000007: E440      in     al,040  ;
00000009: D1E8      shr    ax,1   ;
0000000B: D1E8      shr    ax,1   ;
0000000D: B05C      mov     al,05C  ; '\''
0000000F: 7202      jc    000000013
00000011: B02F      mov     al,02F  ; '/'
;
00000013: B40E      mov     ah,00E
00000015: CD10      int    010
00000017: E2EA      loop   000000003
;
00000019: CD20      int    020

```

83.1.3

MS-DOS, . LODSB DS:SI

[2](#) LODSB .

SCASB .

INT 29h AL.

Peter Ferrie y Andrey «herm1t» Baranovich (11 y 10) [3](#):

Listing 83.1: Andrey «herm1t» Baranovich: 11

```

00000000: B05C      mov     al,05C  ; '\''
;
00000002: AE        scasb
;
00000003: 7A02      jp    000000007
00000005: B02F      mov     al,02F  ; '/'
00000007: CD29      int    029
;
00000009: EBF5      jmp   000000000
;
```

SCASB AL 5Ch AL. JP ..

SALC (AKA SETALC) («Set AL CF»). CPU AL 0xFF CF. 8086/8088.

²<http://go.yurichev.com/17305>

³<http://go.yurichev.com/17087>

Listing 83.2: Peter Ferrie: 10

```
;  
00000000: AE      scasb  
;  
;  
00000001: D6      setalc  
;  
00000002: 242D    and     al,02D ; '-'  
;  
00000004: 042F    add     al,02F ; '/'  
;  
00000006: CD29    int     029      ;  
00000008: EBF6    jmps    000000000 ;
```

.ASCII⁴ («\») 0x5C y 0x2F («/»). CF 0x5C o 0x2F.

AL () 0x2D 0 o 0x2D. 0x2F 0x5C o 0x2F..

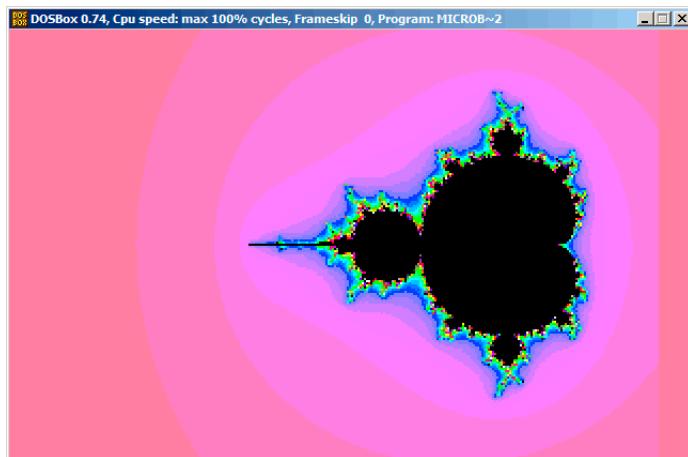
83.1.4 Conclusión

DOSBox, [Windows NT](#) MS-DOS, .

⁴American Standard Code for Information Interchange

5 «Sir_Lagsalot» en 2009, ..

:



$$\bullet : (a + bi) + (c + di) = (a + c) + (b + d)i$$

:

$$\operatorname{Re}(sum) = \operatorname{Re}(a) + \operatorname{Re}(b)$$

$$\operatorname{Im}(sum) = \operatorname{Im}(a) + \operatorname{Im}(b)$$

$$\bullet : (a + bi)(c + di) = (ac - bd) + (bc + ad)i$$

:

$$\operatorname{Re}(product) = \operatorname{Re}(a) \cdot \operatorname{Re}(c) - \operatorname{Re}(b) \cdot \operatorname{Re}(d)$$

$$\operatorname{Im}(product) = \operatorname{Im}(b) \cdot \operatorname{Im}(c) + \operatorname{Im}(a) \cdot \operatorname{Im}(d)$$

$$\bullet : (a + bi)^2 = (a + bi)(a + bi) = (a^2 - b^2) + (2ab)i$$

:

$$\operatorname{Re}(square) = \operatorname{Re}(a)^2 - \operatorname{Im}(a)^2$$

$$\operatorname{Im}(square) = 2 \cdot \operatorname{Re}(a) \cdot \operatorname{Im}(a)$$

$$z_{n+1} = z_n^2 + c \quad (z \text{ y } c \text{ } c).$$

:

• .

• .

• :

- .

- .

- .

- ? .

- .

- .

• ? .

- ?
- .
-

Listing 83.3:

```
def check_if_is_in_set(P):
    P_start=P
    iterations=0

    while True:
        if (P>bounds):
            break
        P=P^2+P_start
        if iterations > max_iterations:
            break
        iterations++

    return iterations

#
for each point on screen P:
    if check_if_is_in_set (P) < max_iterations:

#
for each point on screen P:
    iterations = if check_if_is_in_set (P)
```

Listing 83.4:

```
def check_if_is_in_set(X, Y):
    X_start=X
    Y_start=Y
    iterations=0

    while True:
        if (X^2 + Y^2 > bounds):
            break
        new_X=X^2 - Y^2 + X_start
        new_Y=2*X*Y + Y_start
        if iterations > max_iterations:
            break
        iterations++
```

```

    return iterations

#
for X = min_X to max_X:
    for Y = min_Y to max_Y:
        if check_if_is_in_set (X,Y) < max_iterations:
            X, Y

#
for X = min_X to max_X:
    for Y = min_Y to max_Y:
        iterations = if check_if_is_in_set (X,Y)

        X,Y

```

[6](#), [7](#):

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Mnoj
{
    class Program
    {
        static void Main(string[] args)
        {
            double realCoord, imagCoord;
            double realTemp, imagTemp, realTemp2, arg;
            int iterations;
            for (imagCoord = 1.2; imagCoord >= -1.2; imagCoord ↴
                -= 0.05)
            {
                for (realCoord = -0.6; realCoord <= 1.77; ↴
                    realCoord += 0.03)
                {
                    iterations = 0;
                    realTemp = realCoord;
                    imagTemp = imagCoord;
                    arg = (realCoord * realCoord) + (imagCoord ↴
                        * imagCoord);
                    while ((arg < 2*2) && (iterations < 40))
                    {

```

⁶[wikipedia](#)

⁷: [beginners.re](#)

```
    realTemp2 = (realTemp * realTemp) - (‐  
↳ imagTemp * imagTemp) - realCoord;  
            imagTemp = (2 * realTemp * imagTemp) - ‐  
↳ imagCoord;  
            realTemp = realTemp2;  
            arg = (realTemp * realTemp) + (imagTemp *  
↳ * imagTemp);  
            iterations += 1;  
        }  
        Console.WriteLine("{0,2:D} ", iterations);  
    }  
    Console.WriteLine("\n");  
}  
Console.ReadKey();  
}  
}  
}
```

:
[beginners.re](#).

<http://go.yurichev.com/17309>, ..

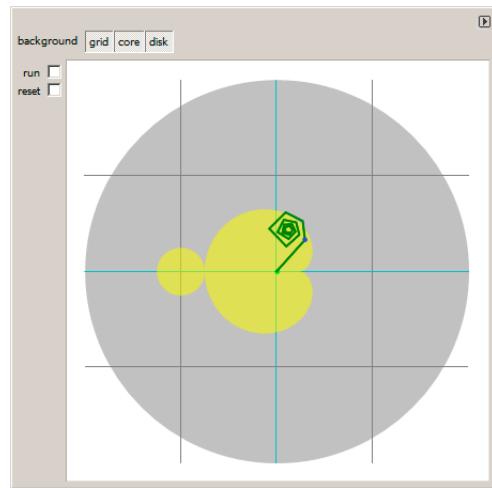


Figura 83.1:

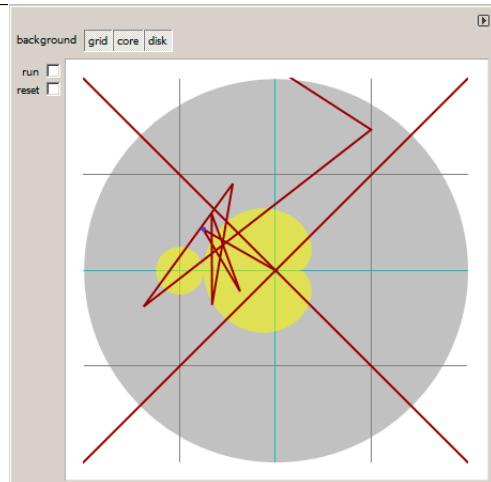


Figura 83.2:

:<http://go.yurichev.com/17310>.

83.2.2

:

Listing 83.5:

```
1 ; X
2 ; Y
3
4
5
6 ; X=0, Y=0           X=319, Y=0
7 ; +----->
8 ; |
9 ;
10 ;
11 ;
12 ;
13 ;
14 ; v
15 ; X=0, Y=199       X=319, Y=199
16
17 ;
18 ;
19 mov al,13h
20 int 10h
21 ;
22 ;
23 ; DS:BX ( DS:0) Program Segment Prefix
24 ; ... CD 20 FF 9F
25 les ax,[bx]
26 ; ES:AX=9FFF:20CD
27
28 FillLoop:
29 ; DX 0. CWD : DX:AX = sign_extend(AX).
30 ;
31 ;
32 cwd
33 mov ax,di
34 ; AX
35 ; 320
36 mov cx,320
37 div cx
38 ; DX (start_X) - (: 0..319); AX - (: 0..199)
39 sub ax,100
40 ; AX=AX-100, AX (start_Y) -100..99
41 ; DX 0..319 0x0000..0x013F
42 dec dh
43 ; DX 0xFF00..0x003F (-256..63)
```

```

44 xor bx,bx
45 xor si,si
46 ; BX (temp_X)=0; SI (temp_Y)=0
47 ;
48 ;
49 ;
50 ;
51 MandelLoop:
52 mov bp,si      ; BP = temp_Y
53 imul si,bx    ; SI = temp_X*temp_Y
54 add si,si     ; SI = SI*2 = (temp_X*temp_Y)*2
55 imul bx,bx    ; BX = BX^2 = temp_X^2
56 jo MandelBreak ; ?
57 imul bp,bp    ; BP = BP^2 = temp_Y^2
58 jo MandelBreak ; ?
59 add bx,bp     ; BX = BX+BP = temp_X^2 + temp_Y^2
60 jo MandelBreak ; ?
61 sub bx,bp     ; BX = BX-BP = temp_X^2 + temp_Y^2 - temp_Y^2 =
62           ↴ temp_X^2
63 sub bx,bp     ; BX = BX-BP = temp_X^2 - temp_Y^2
64 ;
65 sar bx,6      ; BX=BX/64
66 add bx,dx     ; BX=BX+start_X
67 ; temp_X = temp_X^2 - temp_Y^2 + start_X
68 sar si,6      ; SI=SI/64
69 add si,ax     ; SI=SI+start_Y
70 ; temp_Y = (temp_X*temp_Y)*2 + start_Y
71 loop MandelLoop
72
73 MandelBreak:
74 ; CX=
75 xchg ax,cx
76 ; AX=. AL ES:[DI]
77 stosb
78 ; stosb
79 ;
80 ;
81 jmp FillLoop

```

:

- $320 \times 200 \cdot 320 \times 200 = 64000$ (0xFA00). .

ES 0xA000 (PUSH 0A000h / POP ES). : [94 on page 1160](#).

MS-DOS, ...

- . . -100..99 y Y -256..63. . -160..159? SUB DX, 160 , DEC DH2 (0x100 (256) DX). .
 - . .
 - . .
 -
 - . . . -32768..32767,
 -
- MandelBreak,: CX=0 (); . . !
- .
- :
- 1- CWD XOR DX, DX MOV DX, 0.
- 1- XCHG AX, CX MOV AX,CX..
- DI ()⁸ . .

83.2.3

Listing 83.6:

```

1 org 100h
2 mov al,13h
3 int 10h
4
5 ;
6
7 mov dx, 3c8h
8 mov al, 0
9 out dx, al
10 mov cx, 100h
11 inc dx
12 l00:
13 mov al, cl
14 shl ax, 2
15 out dx, al ;
16 out dx, al ;
17 out dx, al ;
18 loop l00
19
20 push 0a000h
21 pop es

```

⁸: <http://go.yurichev.com/17004>

```
22 xor di, di
23
24
25 FillLoop:
26 cwd
27 mov ax,di
28 mov cx,320
29 div cx
30 sub ax,100
31 sub dx,160
32
33 xor bx,bx
34 xor si,si
35
36 MandelLoop:
37 mov bp,si
38 imul si,bx
39 add si,si
40 imul bx,bx
41 jo MandelBreak
42 imul bp,bp
43 jo MandelBreak
44 add bx,bp
45 jo MandelBreak
46 sub bx,bp
47 sub bx,bp
48
49 sar bx,6
50 add bx,dx
51 sar si,6
52 add si,ax
53
54 loop MandelLoop
55
56 MandelBreak:
57 xchg ax,cx
58 stosb
59 cmp di, OFA00h
60 jb FillLoop
61
62 ;
63 xor ax,ax
64 int 16h
65 ;
66 mov ax, 3
67 int 10h
68 ;
```

```
int 20h
```

9

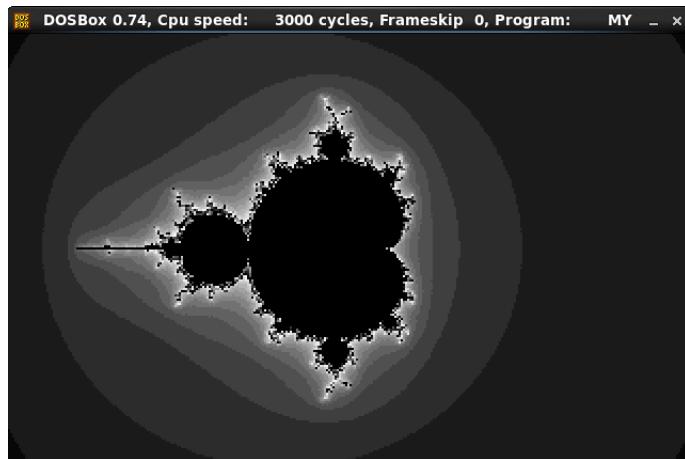


Figura 83.3:

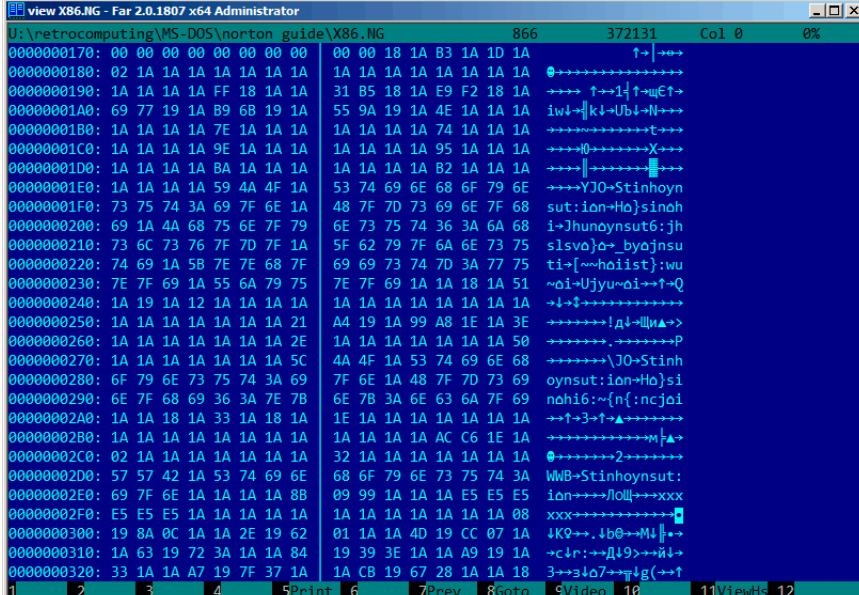
⁹nasm fiole.asm -fbin -o file.com

Parte IX

Ejemplos de ingeniería inversa de formatos de archivo propietarios

Capítulo 84

84.1 Norton Guide:



The screenshot shows a Windows application window titled "view X86.NG - Far 2.0.1807 x64 Administrator". The window displays a file named "X86.NG" located at "U:\retrocomputing\MS-DOS\norton_guide". The file contains a large amount of binary data represented as hex values and their corresponding ASCII characters. The ASCII characters form a guide for using Norton Utilities. The text includes various command names like "FORMAT", "COPY", "MOVE", etc., along with their parameters and descriptions. The window has a standard Windows interface with a menu bar, toolbar, and status bar at the bottom.

Figura 84.1:

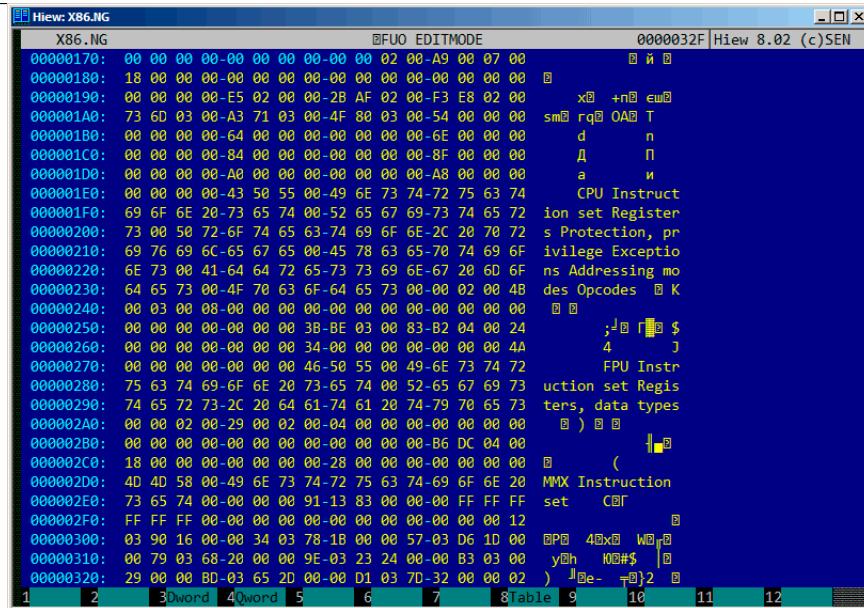


Figura 84.2: Hiew

<http://go.yurichev.com/17317>.

84.1.1

Wolfram Mathematica 10.

Listing 84.1: Wolfram Mathematica 10

```
In[1]:= input = BinaryReadList["X86.NG"];

In[2]:= Entropy[2, input] // N
Out[2]= 5.62724

In[3]:= decrypted = Map[BitXor[#, 16^^1A] &, input];

In[4]:= Export["X86_decrypted.NG", decrypted, "Binary"];

In[5]:= Entropy[2, decrypted] // N
Out[5]= 5.62724

In[6]:= Entropy[2, ExampleData[{"Text", "ShakespearesSonnets"}]] // N
```

```
Out[6]= 4.42366
```

:<http://go.yurichev.com/17350>.

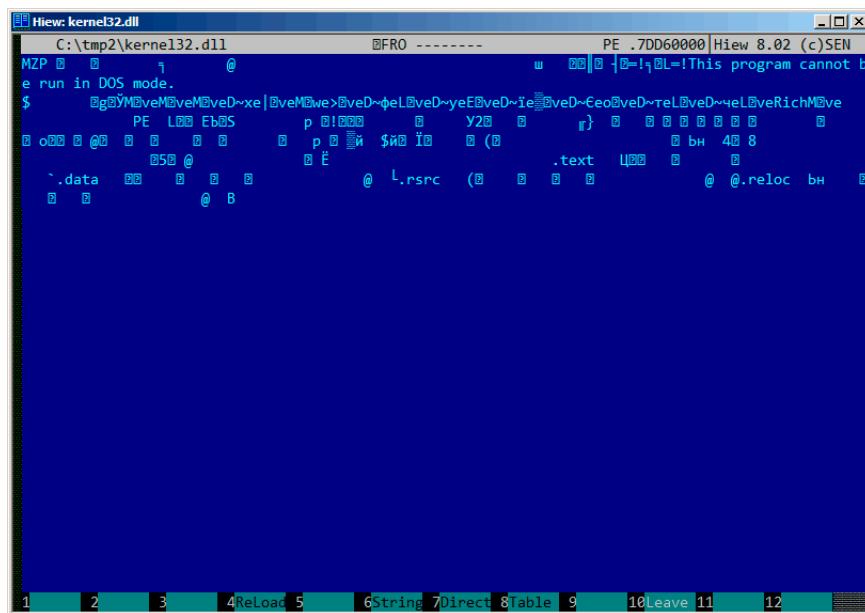


Figura 84.3:

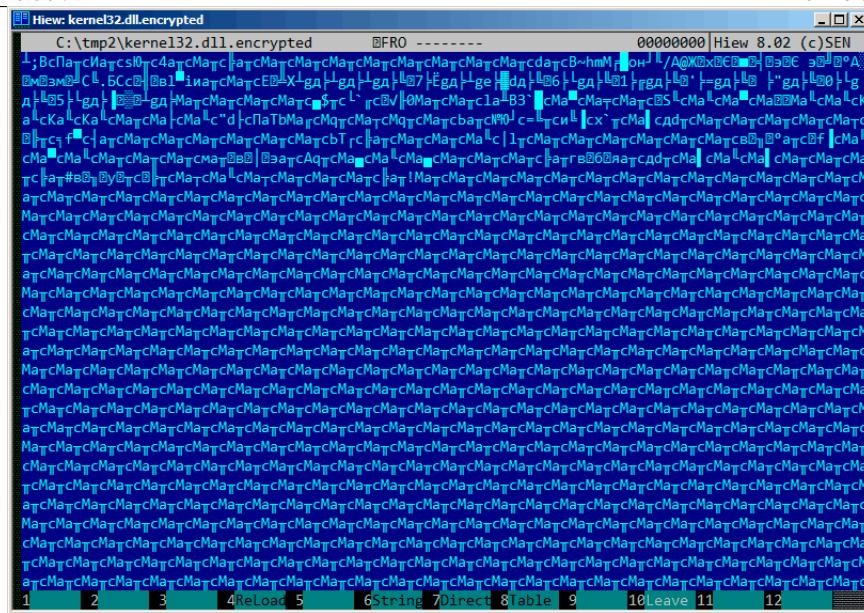


Figura 84.4:



Figura 84.5: PE-

```

Hiew kernel32.dll.encrypted
C:\tmp2\kernel32.dll.encrypted 000000290 Hiew 8.02 (c)SEN
000000E0: 8C 61 D2 63-8C 61 D2 63-DC 24 D2 63-C0 60 D6 63 MatcMatcMatcMatcMatcMatcMatc
000000F0: 09 FB C7 30-8C 61 D2 63-8C 61 D2 63-6C 61 D0 42 MatcMatcMatcMatcMatcMatcMatc
00000100: 87 60 DB 63-8C 61 DF 63-8C 61 D1 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatcMatc
00000110: 1F 53 D3 63-8C 61 D3 63-8C 61 DF 63-8C 61 04 1E MatcMatcMatcMatcMatcMatcMatc
00000120: 8C 61 D3 63-8C 61 D3 63-8A 61 D3 63-8A 61 D3 63 MatcMatcMatcMatcMatcMatcMatc
00000130: 8A 61 D3 63-8C 61 D2 63-8C 61 C3 63-8C 61 D3 63 MatcMatcMatcMatcMatcMatcMatc
00000140: 22 64 C3 63-8F 61 92 62-8C 61 D6 63-8C 71 D2 63 MatcMatcMatcMatcMatcMatcMatc
00000150: 8C 61 C2 63-8C 71 D2 63-8C 61 D2 63-9C 61 D2 63 MatcMatcMatcMatcMatcMatcMatc
00000160: FC 9E D9 63-3D C8 D2 63-A8 C8 DE 63-78 60 D2 63 MatcMatcMatcMatcMatcMatcMatc
00000170: 8C 61 DD 63-A4 64 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatcMatc
00000180: 8C 61 D2 63-8C 61 D2 63-8C 61 C2 63-10 CC D2 63 MatcMatcMatcMatcMatcMatc
00000190: B8 66 DF 63-B4 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
000001A0: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
000001B0: 9C 54 DA 63-CC 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
000001C0: 8C 61 D3 63-7C 6C D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
000001D0: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
000001E0: A2 15 B7 1B-F8 61 D2 63-1A 66 DE 63-8C 61 D3 63 MatcMatcMatcMatcMatcMatc
000001F0: 8C 61 DF 63-8C 61 D3 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
00000200: 8C 61 D2 63-AC 61 D2 03-A2 05 B3 17-ED 61 D2 63 MatcMatcMatcMatcMatcMatc
00000210: 80 71 D2 63-8C 61 DC 63-8C 61 D3 63-8C 61 DC 63 MatcMatcMatcMatcMatcMatc
00000220: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-CC 61 D2 A3 MatcMatcMatcMatcMatcMatc
00000230: A2 13 A1 11-EF 61 D2 63-A4 64 D2 63-8C 61 DD 63 MatcMatcMatcMatcMatcMatc
00000240: 8C 61 D3 63-8C 61 DD 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
00000250: 8C 61 D2 63-CC 61 D2 23-A2 13 B7 0F-E3 02 D2 63 MatcMatcMatcMatcMatcMatc
00000260: 10 CC D2 63-8C 61 C2 63-8C 61 D3 63-8C 61 C2 63 MatcMatcMatcMatcMatcMatc
00000270: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-CC 61 D2 21 MatcMatcMatcMatcMatcMatc
00000280: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc
00000290: 8C 61 D2 63-8C 61 D2 63-8C 61 D2 63-8C 61 D2 63 MatcMatcMatcMatcMatcMatc

```

1Global 2FilBlk 3CryBlk 4ReLoad 5 6String 7Direct 8Table 9 10Leave 11 12AddNam

Figura 84.6:

:8C 61 D2 63.

<http://go.yurichev.com/17352>.

84.2.1 Ejercicio

<http://go.yurichev.com/17353>.

Capítulo 85

«Millenium Return to Earth» ¹.



Figura 85.1:



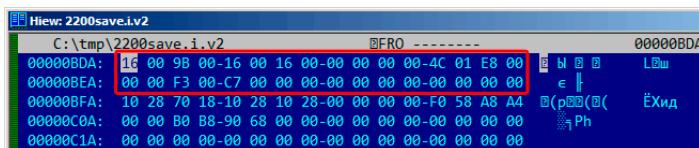
Figura 85.2:

```
...> FC /b 2200save.i.v1 2200SAVE.I.V2
```

```
Comparing files 2200save.i.v1 and 2200SAVE.I.V2
00000016: 0D 04
00000017: 03 04
0000001C: 1F 1E
00000146: 27 3B
00000BDA: 0E 16
00000BDC: 66 9B
00000BDE: 0E 16
00000BE0: 0E 16
00000BE6: DB 4C
00000BE7: 00 01
00000BE8: 99 E8
00000BEC: A1 F3
00000BEE: 83 C7
00000BFB: A8 28
00000BFD: 98 18
00000BFF: A8 28
00000C01: A8 28
00000C07: D8 58
00000C09: E4 A4
00000C0D: 38 B8
00000C0F: E8 68
...
```

.. 0x0E (14) 0xBDA 0x16 (22) .. 0x66 (102) 0xBD_C 0x9B (155) ..

:beginners.re.



The screenshot shows the Hiew debugger interface with the title "Hiew: 2200save.i.v2". The assembly code is displayed in a window titled "C:\tmp\2200save.i.v2". The code is in Intel hex format, showing memory addresses from 00000BDA to 00000C1A. A red box highlights the instruction at address 00000BEA, which is 00 00 F3 00-C7. The assembly code includes labels like FRO, Ldw, e, and Ph. The right side of the screen shows the assembly mnemonics and their corresponding opcodes.

Address	OpCode	Mnemonic	Description
00000BDA	16 00 98 00-16 00 16 00-00 00 00 00-4C 01 E8 00	FRO	Ldw
00000BEA	00 00 F3 00-C7 00 00 00-00 00 00 00-00 00 00 00 00 00	e	
00000BFA	18 28 70 18-10 28 18 28-00 00 00 00-F0 58 A8 A4	(p)(*)(ЕХид
00000C0A	00 00 00 00-90 68 00 00-00 00 00 00-00 00 00 00 00 00	Ph	
00000C1A	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00 00 00		

Figura 85.3: Hiew:

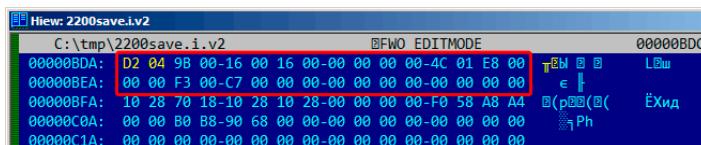


Figura 85.4: Hiew: (0x4D2)



Figura 85.5:

:

Hiew: 2200save.i

C:\DOS\millenium\2200save.i FUO ----- 00000BDA

00000BDA:	FF FF FF FF-FF FF FF FF-FF FF FF FF-FF FF FF FF	00000BDA
00000BEA:	FF FF FF FF-FF FF FF FF-FF FF FF FF-FF FF 00 00	
00000BFA:	10 28 70 18-10 28 10 28-00 00 00 00-F0 58 A8 A4	(p () Ехид
00000C0A:	00 00 B0 B8-90 68 00 00-00 00 00 00-00 00 00 00	Ph
00000C1A:	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	
00000C2A:	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	

Figura 85.6: Hiew:

0xFFFF 65535, :



Figura 85.7: 65535 (0xFFFF)

!:



Figura 85.8:

: 63.4 on page 866.

Capítulo 86

Oracle RDBMS:

----- Call Stack Trace -----			
calling ↳ values in hex location ↳ dubious value)	call type	entry point	argument ↳ (? means ↳)
----- ↳ -----			
_kqvrow()		00000000	
_opifch2() +2729	CALLptr	00000000	23D4B914 ↳
↳ E47F264 1F19AE2			
_kpoal8() +2832	CALLrel	_opifch2()	EB1C8A8 1
_opiindr() +1248	CALLreg	00000000	89 5 EB1CC74
↳ EB1FOAO			5E 1C ↳
_ttcpip() +1051	CALLreg	00000000	5E 1C ↳
↳ EB1FOAO 0			
_opitsk() +1404	CALL???	00000000	C96C040 5E ↳
↳ EB1FOAO 0 EB1ED30			EB1F1CC 53 ↳
↳ E52E 0 EB1F1F8			
_opiino() +980	CALLrel	_opitsk()	0 0
_opiindr() +1248	CALLreg	00000000	3C 4 EB1FBF4
_opidrv() +1201	CALLrel	_opiindr()	3C 4 EB1FBF4 ↳
↳ 0			
_sou2o() +55	CALLrel	_opidrv()	3C 4 EB1FBF4
_opimai_real() +124	CALLrel	_sou2o()	EB1FC04 3C 4 ↳
↳ EB1FBF4			
_opimai() +125	CALLrel	_opimai_real()	2 EB1FC2C
_OracleThreadStart@	CALLrel	_opimai()	2 EB1FF6C 7 ↳
↳ C88A7F4 EB1FC34 0			

4() +830			
77E6481C	CALLreg	00000000	EB1FD04
↳ E41FF9C 0 EB1FFC4			E41FF9C 0 0 ✓
00000000	CALL???	00000000	

. 1 .

Hiew:

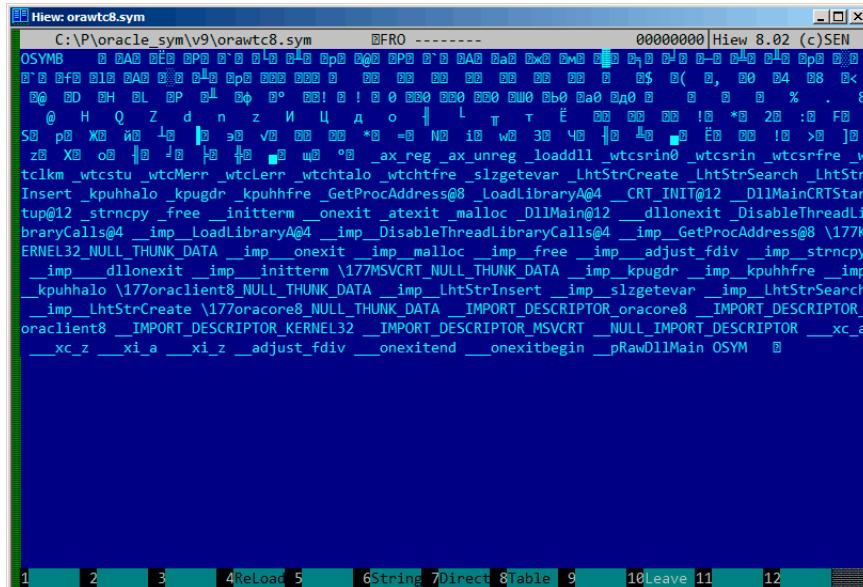


Figura 86.1:

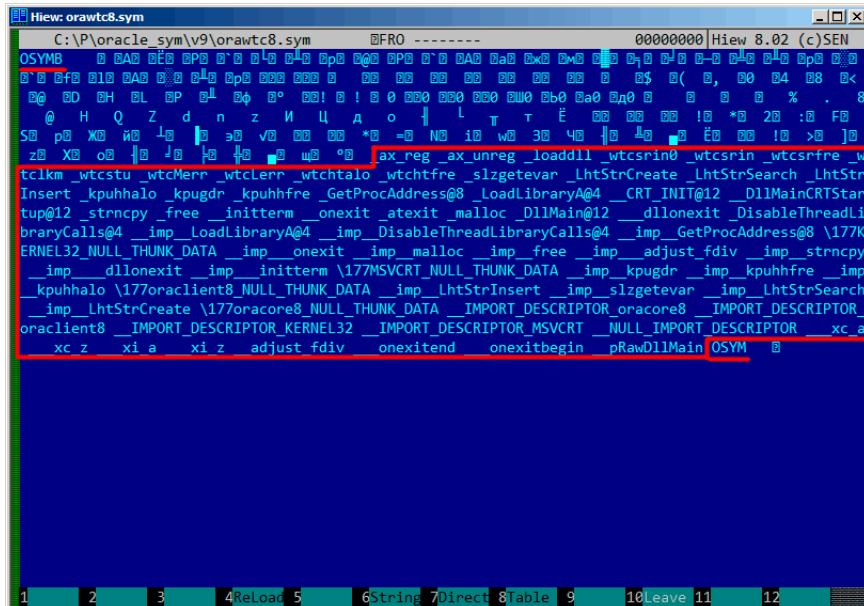


Figura 86.2:

```
strings strings_block | wc -l  
66
```

```
$ hexdump -C orawtc8.sym
00000000  4f 53 59 4d 42 00 00 00  00 10 00 10 80 10 00 10 | ↴
   ↳ OSYMB.....| ↴
00000010  f0 10 00 10 50 11 00 10  60 11 00 10 c0 11 00 10 ↴
   ↳ |....P...`.....| ↴
00000020  d0 11 00 10 70 13 00 10  40 15 00 10 50 15 00 10 ↴
   ↳ |....p@...P....| ↴
00000030  60 15 00 10 80 15 00 10  a0 15 00 10 a6 15 00 10 ↴
   ↳ |`.....| ↴
...
...
```

?

Hiew:

```
Hiew asd2
C:\V\oracle_sym\v9\asd2 FRO -----
00000000: 00 10 00 10-80 10 00 10-F0 10 00 10-50 11 00 10 00 00000000 Hiew 8.02 (c)SEN
00000010: 60 11 00 10-C0 11 00 10-D0 11 00 10-70 13 00 10 00 00000010
00000020: 40 15 00 10-50 15 00 10-60 15 00 10-80 15 00 10 00 00000020
00000030: A0 15 00 10-A6 15 00 10-A5 15 00 10-B2 15 00 10 00 00000030
00000040: B8 15 00 10-BE 15 00 10-C4 15 00 10-CA 15 00 10 00 00000040
00000050: D0 15 00 10-E0 15 00 10-B0 16 00 10-60 17 00 10 00 00000050
00000060: 66 17 00 10-6C 17 00 10-88 17 00 10-B0 17 00 10 00 00000060
00000070: D0 17 00 10-E0 17 00 10-10 18 00 10-16 18 00 10 00 00000070
00000080: 00 20 00 10-04 20 00 10-08 20 00 10-0C 20 00 10 00 00000080
00000090: 10 20 00 10-14 20 00 10-18 20 00 10-1C 20 00 10 00 00000090
000000A0: 20 20 00 10-24 20 00 10-28 20 00 10-2C 20 00 10 00 000000A0
000000B0: 30 20 00 10-34 20 00 10-38 20 00 10-3C 20 00 10 00 000000B0
000000C0: 40 20 00 10-44 20 00 10-48 20 00 10-4C 20 00 10 00 000000C0
000000D0: 50 20 00 10-D0 20 00 10-E4 20 00 10-F8 20 00 10 P 01 0F 00 000000D0
000000E0: 0C 21 00 10-20 21 00 10-00 30 00 10-04 30 00 10 E ! ! 0 000000E0
000000F0: 08 30 00 10-0C 30 00 10-98 30 00 10-9C 30 00 10 00 000000F0
00000100: A0 30 00 10-A4 30 00 10-00 00 00-08 00 00 00 a0 0D0 0 00000100
00000110: 12 00 00 00-1B 00 00 00-25 00 00 00-2E 00 00 00 E % . 00000110
00000120: 38 00 00 00-40 00 00 00-48 00 00 00-51 00 00 00 8 @ H Q 00000120
00000130: 5A 00 00 00-64 00 00 00-6E 00 00 00-7A 00 00 00 Z d n z 00000130
00000140: 88 00 00 00-96 00 00 00-44 00 00 00-4E 00 00 00 И Ц А 00000140
00000150: B6 00 00 00-C0 00 00 00-D2 00 00 00-E2 00 00 00 I L T 00000150
00000160: F0 00 00 00-07 01 00 00-10 01 00 00-16 01 00 00 E 00 00 00 00000160
00000170: 21 01 00 00-2A 01 00 00-32 01 00 00-3A 01 00 00 I8 *8 28 :8 00000170
00000180: 46 01 00 00-53 01 00 00-70 01 00 00-86 01 00 00 F2 S0 p0 X0 00000180
00000190: A9 01 00 00-C1 01 00 00-DE 01 00 00-ED 01 00 00 K0 I0 э0 00000190
000001A0: FB 00 00 00-07 02 00 00-18 02 00 00-2A 02 00 00 VB 00 00 *8 000001A0
1Global 2FileBlk 3CryBlk 4Reload 5String 7Direct 8Table 9Leave 11 12AddNam
```

Figura 86.3:



Figura 86.4:

Listing 86.1:

```
$ od -v -t x4 binary_block
0000000 10001000 10001080 100010f0 10001150
0000020 10001160 100011c0 100011d0 10001370
0000040 10001540 10001550 10001560 10001580
0000060 100015a0 100015a6 100015ac 100015b2
0000100 100015b8 100015be 100015c4 100015ca
0000120 100015d0 100015e0 100016b0 10001760
0000140 10001766 1000176c 10001780 100017b0
0000160 100017d0 100017e0 10001810 10001816
0000200 10002000 10002004 10002008 1000200c
0000220 10002010 10002014 10002018 1000201c
0000240 10002020 10002024 10002028 1000202c
0000260 10002030 10002034 10002038 1000203c
0000300 10002040 10002044 10002048 1000204c
0000320 10002050 100020d0 100020e4 100020f8
0000340 1000210c 10002120 10003000 10003004
```

0000360	10003008	1000300c	10003098	1000309c
0000400	100030a0	100030a4	00000000	00000008
0000420	00000012	0000001b	00000025	0000002e
0000440	00000038	00000040	00000048	00000051
0000460	0000005a	00000064	0000006e	0000007a
0000500	00000088	00000096	000000a4	000000ae
0000520	000000b6	000000c0	000000d2	000000e2
0000540	000000f0	00000107	00000110	00000116
0000560	00000121	0000012a	00000132	0000013a
0000600	00000146	00000153	00000170	00000186
0000620	000001a9	000001c1	000001de	000001ed
0000640	000001fb	00000207	0000021b	0000022a
0000660	0000023d	0000024e	00000269	00000277
0000700	00000287	00000297	000002b6	000002ca
0000720	000002dc	000002f0	00000304	00000321
0000740	0000033e	0000035d	0000037a	00000395
0000760	000003ae	000003b6	000003be	000003c6
0001000	000003ce	000003dc	000003e9	000003f8
0001020				

.?

0x1000.

.text:60351000	sub_60351000	proc near
.text:60351000		
.text:60351000	arg_0	= dword ptr 8
.text:60351000	arg_4	= dword ptr 0Ch
.text:60351000	arg_8	= dword ptr 10h
.text:60351000		
.text:60351000		push ebp
.text:60351001		mov ebp, esp
.text:60351003		mov eax, dword_60353014
.text:60351008		cmp eax, 0FFFFFFFh
.text:6035100B		jnz short loc_6035104F
.text:6035100D		mov ecx, hModule
.text:60351013		xor eax, eax
.text:60351015		cmp ecx, 0FFFFFFFh
.text:60351018		mov dword_60353014, eax
.text:6035101D		jnz short loc_60351031
.text:6035101F		call sub_603510F0
.text:60351024		mov ecx, eax
.text:60351026		mov eax, dword_60353014
.text:6035102B		mov hModule, ecx
.text:60351031		
.text:60351031	loc_60351031:	; CODE ✓
	↳ XREF: sub_60351000+1D	
.text:60351031		test ecx, ecx

.text:60351033	jbe	short loc_6035104F
.text:60351035	push	offset ProcName ; "
↳ ax_reg"		↳
.text:6035103A	push	ecx
↳ hModule		; ↳
.text:6035103B	call	ds:GetProcAddress
...		

, «ax_reg» . «ax_reg».

:

.text:60351080	sub_60351080	proc near
.text:60351080		
.text:60351080	arg_0	= dword ptr 8
.text:60351080	arg_4	= dword ptr 0Ch
.text:60351080		
.text:60351080		push ebp
.text:60351081		mov ebp, esp
.text:60351083		mov eax, dword_60353018
.text:60351088		cmp eax, 0FFFFFFFh
.text:6035108B		jnz short loc_603510CF
.text:6035108D		mov ecx, hModule
.text:60351093		xor eax, eax
.text:60351095		cmp ecx, 0FFFFFFFh
.text:60351098		mov dword_60353018, eax
.text:6035109D		jnz short loc_603510B1
.text:6035109F		call sub_603510F0
.text:603510A4		mov ecx, eax
.text:603510A6		mov eax, dword_60353018
.text:603510AB		mov hModule, ecx
.text:603510B1	loc_603510B1:	
↳ XREF: sub_60351080+1D		; CODE ↳
.text:603510B1		test ecx, ecx
.text:603510B3		jbe short loc_603510CF
.text:603510B5		push offset aAx_unreg ; "
↳ ax_unreg"		↳
.text:603510BA		push ecx
↳ hModule		; ↳
.text:603510BB		call ds:GetProcAddress
...		

? 0x0000? [0...0x3F8]..

```
#include <stdio.h>
```

```
#include <stdint.h>
#include <io.h>
#include <assert.h>
#include <malloc.h>
#include <fcntl.h>
#include <string.h>

int main (int argc, char *argv[])
{
    uint32_t sig, cnt, offset;
    uint32_t *d1, *d2;
    int     h, i, remain, file_len;
    char   *d3;
    uint32_t array_size_in_bytes;

    assert (argv[1]); // file name
    assert (argv[2]); // additional offset (if needed)

    // additional offset
    assert (sscanf (argv[2], "%X", &offset)==1);

    // get file length
    assert ((h=open (argv[1], _O_RDONLY | _O_BINARY, 0))!=
    ↵ !=-1);
    assert ((file_len=lseek (h, 0, SEEK_END))!= -1);
    assert (lseek (h, 0, SEEK_SET)!= -1);

    // read signature
    assert (read (h, &sig, 4)==4);
    // read count
    assert (read (h, &cnt, 4)==4);

    assert (sig==0x4D59534F); // OSYM

    // skip timedatestamp (for 11g)
    // lseek (h, 4, 1);

    array_size_in_bytes=cnt*sizeof(uint32_t);

    // load symbol addresses array
    d1=(uint32_t*)malloc (array_size_in_bytes);
    assert (d1);
    assert (read (h, d1, array_size_in_bytes)==
    ↵ array_size_in_bytes);

    // load string offsets array
    d2=(uint32_t*)malloc (array_size_in_bytes);
```

```

        assert (d2);
        assert (read (h, d2, array_size_in_bytes)==↙
↳ array_size_in_bytes);

        // calculate strings block size
remain=file_len-(8+4)-(cnt*8);

        // load strings block
assert (d3=(char*)malloc (remain));
assert (read (h, d3, remain)==remain);

printf ("#include <idc.idc>\n\n");
printf ("static main() {\n");

        for (i=0; i<cnt; i++)
            printf ("\tMakeName(0x%08X, \"%s\");\n", offset,
↳ + d1[i], &d3[d2[i]]);

        printf ("}\n");

        close (h);
        free (d1); free (d2); free (d3);
};

}

```

:

```

#include <idc.idc>

static main() {
    MakeName(0x60351000, "_ax_reg");
    MakeName(0x60351080, "_ax_unreg");
    MakeName(0x603510F0, "_loaddll");
    MakeName(0x60351150, "_wtcsrin0");
    MakeName(0x60351160, "_wtcsrin");
    MakeName(0x603511C0, "_wtcsrfre");
    MakeName(0x603511D0, "_wtclkm");
    MakeName(0x60351370, "_wtcstu");
    ...
}

```

:beginners.re.

. ?

•
•

```
Hiew: oracle.sym
oracle.sym FRO -----
00000000: 4F 53 59 4D-41 4D 36 34-BD 6D 05 00-00 00 00 00 [SYMAM64.m@]
00000010: CD 21 2A 47-00 00 00 00-00 00 00 00 00 00 00 00 =!*G
00000020: 00 00 00 00-00 00 00 00-00 00 40 00-00 00 00 00 @
00000030: 00 10 40 00-00 00 00 00-0C 10 40 00-00 00 00 00 10@_
00000040: 04 11 40 00-00 00 00 00-00-80 13 40 00-00 00 00 00 AB@_
00000050: E3 13 40 00-00 00 00 00-01 14 40 00-00 00 00 00 y@_B@_
00000060: 1F 14 40 00-00 00 00 00-00-3E 14 40 00-00 00 00 00 >B@_
00000070: 54 14 40 00-00 00 00 00-00-1E 18 40 00-00 00 00 00 T@_B@_
00000080: 97 1B 40 00-00 00 00 00-C1 1B 40 00-00 00 00 00 1@_B@_
00000090: 0A 1C 40 00-00 00 00 00-00-4C 1C 40 00-00 00 00 00 L@_B@_
000000A0: 7A 1C 40 00-00 00 00 00-00-98 1C 40 00-00 00 00 00 z@_B@_
000000B0: E7 25 40 00-00 00 00 00-00-11 26 40 00-00 00 00 00 R@_B@_
000000C0: 88 26 40 00-00 00 00 00-00-C4 26 40 00-00 00 00 00 A@_B@_
000000D0: F4 26 40 00-00 00 00 00-00-24 27 40 00-00 00 00 00 I@_B@_
000000E0: 50 27 40 00-00 00 00 00-00-78 27 40 00-00 00 00 00 P@_B@_
000000F0: A0 27 40 00-00 00 00 00-00-4E 28 40 00-00 00 00 00 a@_B@_
00000100: 26 29 40 00-00 00 00 00-00-B4 2C 40 00-00 00 00 00 ;@_B@_
00000110: 66 2D 40 00-00 00 00 00-00-A6 2D 40 00-00 00 00 00 f@_B@_
00000120: 30 2E 40 00-00 00 00 00-00-BA 2E 40 00-00 00 00 00 .@_B@_
00000130: F2 30 40 00-00 00 00 00-00-84 31 40 00-00 00 00 00 C@_B@_
00000140: F0 31 40 00-00 00 00 00-00-5E 32 40 00-00 00 00 00 E@_B@_
00000150: CC 32 40 00-00 00 00 00-00-3A 33 40 00-00 00 00 00 ;@_B@_
00000160: A8 33 40 00-00 00 00 00-00-16 34 40 00-00 00 00 00 i@_B@_
00000170: 84 34 40 00-00 00 00 00-F2 34 40 00-00 00 00 00 D@_B@_
00000180: 60 35 40 00-00 00 00 00-00-CC 35 40 00-00 00 00 00 5@_B@_
00000190: 3A 36 40 00-00 00 00 00-00-A8 36 40 00-00 00 00 00 :@_B@_
000001A0: 16 37 40 00-00 00 00 00-00-84 37 40 00-00 00 00 00 7@_B@_
1Global 2FileBlk 3CryBlk 4Reload 5String 7Direct 8Table 9Leave 11AddNam
```

Figura 86.5:

! : GitHub.

Capítulo 87

Oracle RDBMS: Archivos .MSB

.

1.

ORAUS.MSG :

Listing 87.1: ORAUS.MSG

```
00000, 00000, "normal, successful completion"
00001, 00000, "unique constraint (%s.%s) violated"
00017, 00000, "session requested to set trace event"
00018, 00000, "maximum number of sessions exceeded"
00019, 00000, "maximum number of session licenses exceeded"
00020, 00000, "maximum number of processes (%s) exceeded"
00021, 00000, "session attached to some other process; cannot ↴
    switch session"
00022, 00000, "invalid session ID; access denied"
00023, 00000, "session references process private memory; ↴
    cannot detach session"
00024, 00000, "logins from more than one process not allowed in ↴
    single-process mode"
00025, 00000, "failed to allocate %s"
00026, 00000, "missing or invalid session ID"
00027, 00000, "cannot kill current session"
00028, 00000, "your session has been killed"
00029, 00000, "session is not a user session"
00030, 00000, "User session ID does not exist."
00031, 00000, "session marked for kill"
...

```

..

ORAUS.MSB . :

```

Hiew: oraus.msb
C:\tmp\oraus.msb          DFR0 ----- 000013B9 | Hiew 8.02 (c)SEN
[...]

```

The text content of the oraus.msb file is as follows:

Normal, successful completionunique constraint (%s.%s) violatedsession requested to set trace eventmaximum number of sessions exceededmaximum number of sessions exceededmaximum number of sessions exceededmaximum number of processes (%s) exceededsession attached to some other process; cannot switch sessioninvalid session ID; access deniedsession references process private memory; cannot detach sessionlogins from more than one process not allowed in single-process modeP %e B %E % { % r % i % ! .%" Z# }%\$ | % failed to allocate %missing or invalid session IDcannot Kill current sessionyour session has been killedsession is not a user sessionUser session ID does not exist.session marked for killinvalid session migration passwordcurrent session has empty migration passwordcannot %s in current PL/SQL sessionLICENSE_MAX_USERS cannot be less than current number of usersmaximum number of recursive SQL levels (%s) exceeded
D & Д'я() %* 3+ J02 aB3 b04 % .Cannot switch to a session belonging to a different server groupCannot create session: server group belongs to another user error during periodic actionactive time limit exceeded - call abortactive time limit exceeded - session terminatedUnknown Service name %sremote operation failedoperating system error occurred while obtaining an enqueuetimeout occurred while waiting for a resourcemaximum number of enqueue resources (%s) exceeded %5 > 6 a7 r8 %9 %: 98; q%< %@= % maximum number of enqueues exceededresource busy and acquire with NOWAIT specified or timeout expiredmaximum number of DML locks exceededDDL lock on object '%s.%s' is already held in an incompatible modemaximum number of temporary table locks exceededDB_BLOCK_SIZE must be %s to mount this database (not %s)maximum number of DB_FILES exceededdeadlock detected while waiting for resourceanother instance has a different DML_LOCKS setting % > D ? z @ 6 A % C . D 00E 10F K0G %H
% DML full-table lock cannot be acquired; DML_LOCKS is %maximum number of log files exceeded %sobject is too large to allocate on this O/S (%s,%s,%s)initialization of FIXED_DATE failedinvalid value %s for parameter %s; must be at least %sinvalid value %s for parameter %s, must be between %s and %scannot acquire lock -- table locks disabled for %scommand %s is not validprocess number must be between 1 and %sprocess "%s" is not active % I P J ~ K W L T M L N 40
% D P % B R h0S w0T % E command %s takes between %s and %s argument(s)no process has
1 2 3 4 Reload 5 6 String 7 Direct 8 Table 9 10 Leave 11 12

Figura 87.1: Hiew:

:

C:\tmp\ora.us.msb 0FRO ----- 00001400 Hiew 8.02 (c)SEN

00001400:	0A 00 00 00-00 00 44 00-01 00 00 00-61 00 11 00	D a
00001410:	00 00 83 00-12 00 00 00-A7 00 13 00-00 00 CA 00	Г з ы ы
00001420:	14 00 00-00-F5 00 15 00-00 00 1E 01-16 00 00 00	и ы ы ы
00001430:	5B 01 17 00-00 00 7C 01-18 00 00 00-B0 01 00 00	[] []]
00001440:	00 00 00 02-6E 6F 72 6D-61 6C 2C 20-73 75 63 63	Normal, succ
00001450:	65 73 73 66-75 6C 20 63-6F 6D 70 6C-65 74 69 6F	essful completio
00001460:	6E 75 6E 69-71 75 65 20-63 6F 6E 73-74 72 61 69	nique constrai
00001470:	6E 74 20 28-25 73 2E 25-73 29 20 76-69 6F 6C 61	nt (%s.%s) viola
00001480:	74 65 64 73-65 73 73 69-6F 6E 20 72-65 71 75 65	tedsession requie
00001490:	73 74 65 64-20 74 6F 20-73 65 74 20-74 72 61 63	sted to set trac
000014A0:	65 20 65 76-65 6E 74 60-61 78 69 6D-75 6D 20 6E	e eventsmaximum n
000014B0:	75 6D 62 65-72 20 6F 66-20 73 65 73 69 6F 6E	umber of session
000014C0:	73 20 65 78-63 65 65 64-65 64 60 61-78 69 6D 75	s exceededmaximum
000014D0:	6D 20 6E 75-6D 62 65 72-20 6F 66 20-73 65 73 73	m number of sess
000014E0:	69 6F 6E 20-6C 69 63 65-6E 73 65 73-20 65 78 63	ion licenses exc
000014F0:	65 65 64 65-64 6D 61 78-69 6D 75 6D-20 6E 75 6D	eededmaximum num
00001500:	62 65 72 20-6F 66 20 70-72 6F 63 65-73 73 65 73	ber of processes
00001510:	20 28 25 73-29 20 65 78-63 65 65 64-65 64 73 65	(%s) exceededse
00001520:	73 73 69 6F-6E 20 61 74-74 61 63 68-65 64 20 74	ssion attached t
00001530:	6F 20 73 6F-6D 65 20 6F-74 68 65 72-20 70 72 6F	o some other pro
00001540:	63 65 73 73-3B 20 63 61-6E 6E 6F 74-20 73 77 69	cess; cannot swi
00001550:	74 63 68 20-73 65 73 73-69 6F 6E 69-6F 76 61 6C	tch sessioninval
00001560:	69 64 20 73-65 73 73 69-6F 6E 20 49-44 3B 20 61	id session ID; a
00001570:	63 63 65 73-73 20 64 65-6E 69 65 64-73 65 73 73	ccess deniedsess
00001580:	69 6F 6E 20-72 65 66 65-72 65 6E 63-65 73 20 70	ion references p
00001590:	72 6F 63 65-73 73 20 70-72 69 76 61-74 65 20 6D	rocess private m
000015A0:	65 6D 6F 72-79 3B 20 63-61 6E 6E 6F-74 20 64 65	emory; cannot de
000015B0:	74 61 63 68-20 73 65 73-73 69 6F 6E-6C 6F 67 69	tach sessionlogi

Figura 87.2: Hiew:

..... ORAUS.MSG : 0, 1, 17 (0x11), 18 (0x12), 19 (0x13), 20 (0x14), 21 (0x15), 22 (0x16), 23 (0x17), 24 (0x18).....

...

«normal, successful completion» 29 (0x1D) . «unique constraint (%s.%s) violated» 34 (0x22) . (0x1D o/y 0x22) .

. Oracle RDBMS ? «normal, successful completion» 0x1444 () 0x44 (). «unique constraint (%s.%s) violated» 0x1461 () 0x61 (). (0x44 y 0x61) !.

:

- 16-;
- 16-;
- 16-.

. ? !!.

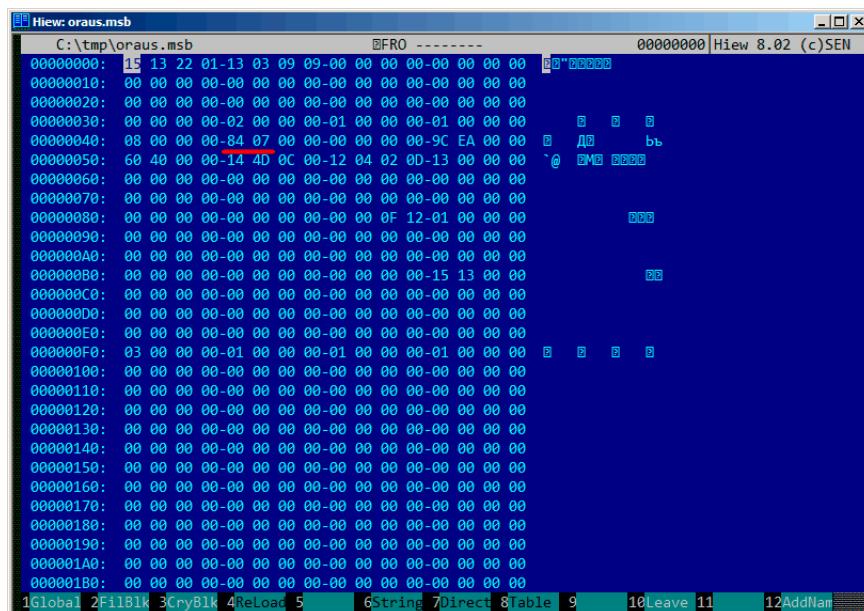


Figura 87.3: Hiew:

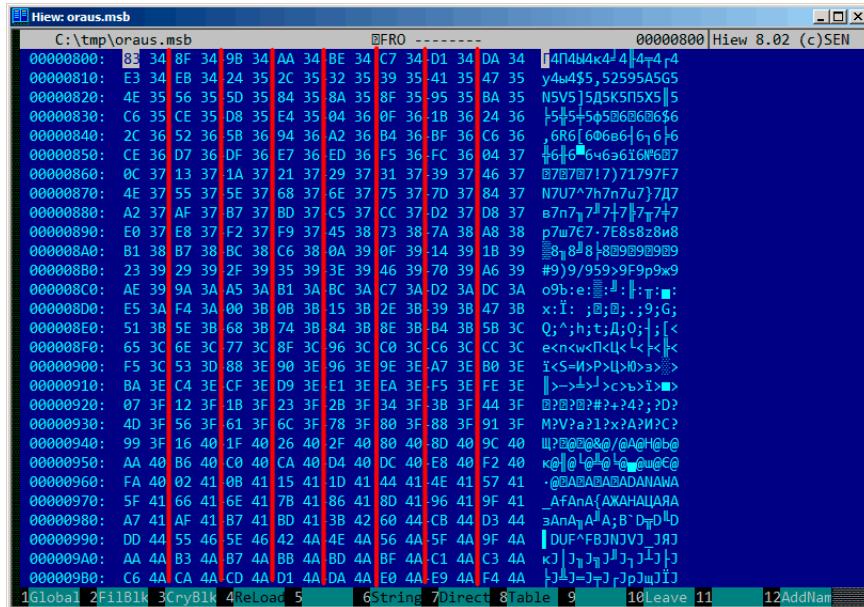


Figura 87.4: Hiew: last_errno

- last_errno();
- ;
- ;
- ;
- ;
- ;

:beginners.re.

(Oracle RDBMS 11.1.0.6): beginners.re, beginners.re.

87.1

Parte X

Capítulo 88

npad

listing.inc (MSVC):

```
; ; LISTING.INC
; ;
; ; This file contains assembler macros and is included by the ↵
; ; files created
; ; with the -FA compiler switch to be assembled by MASM ( ↵
; ; Microsoft Macro
; ; Assembler).
; ;
; ; Copyright (c) 1993-2003, Microsoft Corporation. All rights ↵
; ; reserved.

; ; non destructive nops
npad macro size
if size eq 1
    nop
else
    if size eq 2
        mov edi, edi
    else
        if size eq 3
            ; lea ecx, [ecx+00]
            DB 8DH, 49H, 00H
        else
            if size eq 4
                ; lea esp, [esp+00]
                DB 8DH, 64H, 24H, 00H
            else
                if size eq 5
```

```

    add eax, DWORD PTR 0
else
if size eq 6
; lea ebx, [ebx+00000000]
DB 8DH, 9BH, 00H, 00H, 00H, 00H
else
if size eq 7
; lea esp, [esp+00000000]
DB 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H
else
if size eq 8
; jmp .+8; .npad 6
DB 0EBH, 06H, 8DH, 9BH, 00H, 00H, 00H, 00H
else
if size eq 9
; jmp .+9; .npad 7
DB 0EBH, 07H, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H
else
if size eq 10
; jmp .+A; .npad 7; .npad 1
DB 0EBH, 08H, 8DH, 0A4H, 24H, 00H, 00H, 00H, 90H
else
if size eq 11
; jmp .+B; .npad 7; .npad 2
DB 0EBH, 09H, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H, 82H, 0FFH
else
if size eq 12
; jmp .+C; .npad 7; .npad 3
DB 0EBH, 0AH, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H, 82H, 49H, 00H
else
if size eq 13
; jmp .+D; .npad 7; .npad 4
DB 0EBH, 0BH, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H, 8DH, 64H, 24H, 00H
else
if size eq 14
; jmp .+E; .npad 7; .npad 5
DB 0EBH, 0CH, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H, 05H, 00H, 00H, 00H, 00H
else
if size eq 15
; jmp .+F; .npad 7; .npad 6
DB 0EBH, 0DH, 8DH, 0A4H, 24H, 00H, 00H, 00H, 00H, 8DH, 9BH, 00H, 00H, 00H, 00H
else

```

```
%out error: unsupported npad size
    .err
endif
endifm
```

Capítulo 89

89.1

.....

.

89.2 x86

:

- . 0x90 ([NOP](#)).
- 74 xx (JZ),. (*jump offset*).
- : 0xEB JMP.
- RETN (0xC3) . stdcall ([64.2 on page 869](#)). stdcall RETN (0xC2).
- 0 o 1. MOV EAX, 0 o MOV EAX, 1,. XOR EAX, EAX (2 0x31 0xC0) o XOR EAX, EAX / INC EAX (3 0x31 0xC0 0x40).

.....

.

([68.2.6 on page 903](#)) (: Spanish text placeholder [7.12](#)).

Capítulo 90

Compiler intrinsic

: [MSDN](#).

Capítulo 91

Listing 91.1: libserver11.a

.text:08114CF1	loc_8114CF1: ; CODE XREF: ↴
↳ __PGOSF539_kdlimemSer+89A	
.text:08114CF1	; ↴
↳ __PGOSF539_kdlimemSer+3994	
.text:08114CF1 8B 45 08	mov eax, [ebp+arg_0]
.text:08114CF4 0F B6 50 14	movzx edx, byte ptr [eax, ↴ +14h]
.text:08114CF8 F6 C2 01	test dl, 1
.text:08114CFB 0F 85 17 08 00 00	jnz loc_8115518
.text:08114D01 85 C9	test ecx, ecx
.text:08114D03 0F 84 8A 00 00 00	jz loc_8114D93
.text:08114D09 0F 84 09 08 00 00	jz loc_8115518
.text:08114D0F 8B 53 08	mov edx, [ebx+8]
.text:08114D12 89 55 FC	mov [ebp+var_4], edx
.text:08114D15 31 C0	xor eax, eax
.text:08114D17 89 45 F4	mov [ebp+var_C], eax
.text:08114D1A 50	push eax
.text:08114D1B 52	push edx
.text:08114D1C E8 03 54 00 00	call len2nbytes
.text:08114D21 83 C4 08	add esp, 8

Listing 91.2:

.text:0811A2A5	loc_811A2A5: ; CODE XREF: ↴
↳ kdliSerLengths+11C	
.text:0811A2A5	; kdliSerLengths ↴
↳ +1C1	
.text:0811A2A5 8B 7D 08	mov edi, [ebp+arg_0]
.text:0811A2A8 8B 7F 10	mov edi, [edi+10h]
.text:0811A2AB 0F B6 57 14	movzx edx, byte ptr [edi, ↴ +14h]
.text:0811A2AF F6 C2 01	test dl, 1

.text:0811A2B2 75 3E	jnz	short loc_811A2F2
.text:0811A2B4 83 E0 01	and	eax, 1
.text:0811A2B7 74 1F	jz	short loc_811A2D8
.text:0811A2B9 74 37	jz	short loc_811A2F2
.text:0811A2BB 6A 00	push	0
.text:0811A2BD FF 71 08	push	dword ptr [ecx+8]
.text:0811A2C0 E8 5F FE FF FF	call	len2nbytes

:19.2.4 on page 401, 39.3 on page 623, 47.7 on page 680, 18.7 on page 367, 12.4.1 on page 174, 19.5.2 on page 422.

Capítulo 92

OpenMP

OpenMP .

. . «proof of work»¹ ().

Bitcoin, «hello, world!_» «hello, world!_<number>».

0..INT32_MAX-1 (, 0x7FFFFFFE o 2147483646).

:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include "sha512.h"

int found=0;
int32_t checked=0;

int32_t* __min;
int32_t* __max;

time_t start;

#ifndef __GNUC__
#define min(X,Y) ((X) < (Y) ? (X) : (Y))
#define max(X,Y) ((X) > (Y) ? (X) : (Y))
#endif

void check_nonce (int32_t nonce)
```

```

{

    uint8_t buf[32];
    struct sha512_ctx ctx;
    uint8_t res[64];

    // update statistics
    int t=omp_get_thread_num();

    if (__min[t]==-1)
        __min[t]=nonce;
    if (__max[t]==-1)
        __max[t]=nonce;

    __min[t]=min(__min[t], nonce);
    __max[t]=max(__max[t], nonce);

    // idle if valid nonce found
    if (found)
        return;

    memset (buf, 0, sizeof(buf));
    sprintf (buf, "hello, world!_%d", nonce);

    sha512_init_ctx (&ctx);
    sha512_process_bytes (buf, strlen(buf), &ctx);
    sha512_finish_ctx (&ctx, &res);
    if (res[0]==0 && res[1]==0 && res[2]==0)
    {
        printf ("found (thread %d): [%s]. seconds spent=%d\n",
            t, buf, time(NULL)-start);
        found=1;
    };
    #pragma omp atomic
    checked++;

    #pragma omp critical
    if ((checked % 100000)==0)
        printf ("checked=%d\n", checked);
};

int main()
{
    int32_t i;
    int threads=omp_get_max_threads();
    printf ("threads=%d\n", threads);

    __min=(int32_t*)malloc(threads*sizeof(int32_t));
}

```

```

__max=(int32_t*)malloc(threads*sizeof(int32_t));
for (i=0; i<threads; i++)
    __min[i]=__max[i]=-1;

start=time(NULL);

#pragma omp parallel for
for (i=0; i<INT32_MAX; i++)
    check_nonce (i);

for (i=0; i<threads; i++)
    printf ("__min[%d]=0x%08x __max[%d]=0x%08x\n",
        i, __min[i], i, __max[i]);

free(__min); free(__max);
};

```

check_nonce() .

:

```

#pragma omp parallel for
for (i=0; i<INT32_MAX; i++)
    check_nonce (i);

```

#pragma check_nonce() INT32_MAX(0x7fffffff o 2147483647). #pragma,

²

³ en MSVC 2012:

```

cl openmp_example.c sha512.obj /openmp /O1 /Zi /c
    ↴ Faopenmp_example.asm

```

GCC:

```

gcc -fopenmp 2.c sha512.c -S -masm=intel

```

92.1 MSVC

MSVC 2012 :

Listing 92.1: MSVC 2012

push	OFFSET _main\$omp\$1
------	----------------------

²N.B.:

³sha512.(c|h) y u64.h :<http://go.yurichev.com/17324>

```

push    0
push    1
call    __vcomp_fork
add    esp, 16
; ↴
↳ 00000010H

```

vcomp OpenMP vcomp*.dll ..

_main\$omp\$1:

Listing 92.2: MSVC 2012

```

$T1 = -8 ; size ↴
↳ = 4
$T2 = -4 ; size ↴
↳ = 4
_main$omp$1 PROC ; ↴
↳ COMDAT
    push    ebp
    mov     ebp, esp
    push    ecx
    push    ecx
    push    esi
    lea     eax, DWORD PTR $T2[ebp]
    push    eax
    lea     eax, DWORD PTR $T1[ebp]
    push    eax
    push    1
    push    1
    push    2147483646 ; 7 ↴
    ↳ ffffffeH
    push    0
    call    __vcomp_for_static_simple_init
    mov     esi, DWORD PTR $T1[ebp]
    add    esp, 24 ; ↴
    ↳ 00000018H
    jmp     SHORT $LN6@main$omp$1
$LL2@main$omp$1:
    push    esi
    call    _check_nonce
    pop     ecx
    inc     esi
$LN6@main$omp$1:
    cmp     esi, DWORD PTR $T2[ebp]
    jle     SHORT $LL2@main$omp$1
    call    __vcomp_for_static_end
    pop     esi
    leave

```

```
    ret      0
_main$omp$1 ENDP
```

n n. vcomp_for_static_simple_init() for() . \$T1 y \$T2. 7fffffffh
(o 2147483646) vcomp_for_static_simple_init() .

check_nonce() .

check_nonce() .

:

threads=4

...

checked=2800000

checked=3000000

checked=3200000

checked=3300000

found (thread 3): [hello, world!_1611446522]. seconds spent=3

__min[0]=0x00000000 __max[0]=0xffffffff

__min[1]=0x20000000 __max[1]=0x3fffffff

__min[2]=0x40000000 __max[2]=0x5fffffff

__min[3]=0x60000000 __max[3]=0x7fffffff

:

C:\...\\sha512sum test

000000 ↵

↳ f4a8fac5a4ed38794da4c1e39f54279ad5d9bb3c5465cdf57adaf60403 ↵

↳

df6e3fe6019f5764fc9975e505a7395fed780fee50eb38dd4c0279cb114672e2 ↵

↳ *test

task manager : ... CPU OpenMP .

..

.

:

```
#pragma omp atomic
checked++;

#pragma omp critical
if ((checked % 100000)==0)
    printf ("checked=%d\n", checked);
```

Listing 92.3: MSVC 2012

```

    push    edi
    push    OFFSET _checked
    call    __vcomp_atomic_add_i4
; Line 55
    push    OFFSET _$vcomp$critsect$
    call    __vcomp_enter_critsect
    add    esp, 12
    ↳ 0000000cH ; ↴
; Line 56
    mov     ecx, DWORD PTR _checked
    mov     eax, ecx
    cdq
    mov     esi, 100000
    ↳ 000186a0H ; ↴
    idiv    esi
    test    edx, edx
    jne     SHORT $LN1@check_nonc
; Line 57
    push    ecx
    push    OFFSET ??_C@_0M@NPNHLI00@checked?DN?$CFd?6??
    ↳ $AA@
    call    _printf
    pop     ecx
    pop     ecx
$LN1@check_nonc:
    push    DWORD PTR _$vcomp$critsect$
    call    __vcomp_leave_critsect
    pop     ecx

```

vcomp_atomic_add_i4() vcomp*.dll LOCK XADD⁴.

vcomp_enter_critsect() EnterCriticalSection() ⁵.

92.2 GCC

GCC 4.8.1 , .

Listing 92.4: GCC 4.8.1

```

    mov     edi, OFFSET FLAT:main._omp_fn.0
    call    GOMP_parallel_start
    mov     edi, 0
    call    main._omp_fn.0

```

⁴: A.6.1 on page 1231

⁵: 68.4 on page 938

	call GOMP_parallel_end
--	------------------------

..
main._omp_fn.0:

Listing 92.5: GCC 4.8.1

```

main._omp_fn.0:
    push    rbp
    mov     rbp,  rsp
    push    rbx
    sub     rsp,  40
    mov     QWORD PTR [rbp-40], rdi
    call    omp_get_num_threads
    mov     ebx,  eax
    call    omp_get_thread_num
    mov     esi,  eax
    mov     eax,  2147483647 ; 0xFFFFFFFF
    cdq
    idiv   ebx
    mov     ecx,  eax
    mov     eax,  2147483647 ; 0xFFFFFFFF
    cdq
    idiv   ebx
    mov     eax,  edx
    cmp     esi,  eax
    jl     .L15

.L18:
    imul   esi,  ecx
    mov     edx,  esi
    add     eax,  edx
    lea    ebx,  [rax+rcx]
    cmp     eax,  ebx
    jge    .L14
    mov     DWORD PTR [rbp-20], eax

.L17:
    mov     eax, DWORD PTR [rbp-20]
    mov     edi,  eax
    call    check_nonce
    add     DWORD PTR [rbp-20], 1
    cmp     DWORD PTR [rbp-20], ebx
    jl     .L17
    jmp    .L14

.L15:
    mov     eax,  0
    add     ecx,  1
    jmp    .L18

```

.L14:

```
add    rsp, 40
pop    rbx
pop    rbp
ret
```

omp_get_num_threads() y omp_get_thread_num() ,. check_nonce().

GCC LOCK ADD :

Listing 92.6: GCC 4.8.1

```
lock add    DWORD PTR checked[rip], 1
call    GOMP_critical_start
mov     ecx, DWORD PTR checked[rip]
mov     edx, 351843721
mov     eax, ecx
imul    edx
sar     edx, 13
mov     eax, ecx
sar     eax, 31
sub     edx, eax
mov     eax, edx
imul    eax, eax, 100000
sub     ecx, eax
mov     eax, ecx
test    eax, eax
jne    .L7
mov     eax, DWORD PTR checked[rip]
mov     esi, eax
mov     edi, OFFSET FLAT:.LC2 ; "checked=%d\n"
mov     eax, 0
call    printf
.L7:
call    GOMP_critical_end
```

GOMP GNU OpenMP. vcomp*.dll, : [GitHub](#).

Capítulo 93

Itanium

.

:

Listing 93.1: Linux kernel 3.2.0.4

```
#define TEA_ROUNDS          32
#define TEA_DELTA           0x9e3779b9

static void tea_encrypt(struct crypto_tfm *tfm, u8 *dst, const u8 *src)
{
    u32 y, z, n, sum = 0;
    u32 k0, k1, k2, k3;
    struct tea_ctx *ctx = crypto_tfm_ctx(tfm);
    const __le32 *in = (const __le32 *)src;
    __le32 *out = (__le32 *)dst;

    y = le32_to_cpu(in[0]);
    z = le32_to_cpu(in[1]);

    k0 = ctx->KEY[0];
    k1 = ctx->KEY[1];
    k2 = ctx->KEY[2];
    k3 = ctx->KEY[3];

    n = TEA_ROUNDS;

    while (n-- > 0) {
        sum += TEA_DELTA;
```

```

        ↵ k1);
        ↵ z += ((y << 4) + k2) ^ (y + sum) ^ ((y >> 5) + ↵
        ↵ k3);
    }

    out[0] = cpu_to_le32(y);
    out[1] = cpu_to_le32(z);
}

```

:

Listing 93.2: Linux Kernel 3.2.0.4 Itanium 2 (McKinley)

0090	tea_encrypt:
0090 08 80 80 41 00 21	adds r16 = 96, r32 ↵
↳ // ptr to ctx->KEY[2]	
0096 80 C0 82 00 42 00	adds r8 = 88, r32 ↵
↳ // ptr to ctx->KEY[0]	
009C 00 00 04 00	nop.i 0
00A0 09 18 70 41 00 21	adds r3 = 92, r32 ↵
↳ // ptr to ctx->KEY[1]	
00A6 F0 20 88 20 28 00	ld4 r15 = [r34], 4 ↵
↳ // load z	
00AC 44 06 01 84	adds r32 = 100, r32 ↵
↳ ; // ptr to ctx->KEY[3]	
00B0 08 98 00 20 10 10	ld4 r19 = [r16] ↵
↳ // r19=k2	
00B6 00 01 00 00 42 40	mov r16 = r0 ↵
↳ // r0 always contain zero	
00BC 00 08 CA 00	mov.i r2 = ar.lc ↵
↳ // save lc register	
00C0 05 70 00 44 10 10 9E FF FF FF FF 7F 20	ld4 r14 = [r34] ↵
↳ // load y	
00CC 92 F3 CE 6B	movl r17 = 0 ↵
↳ xFFFFFFFF9E3779B9;; // TEA_DELTA	
00D0 08 00 00 00 01 00	nop.m 0
00D6 50 01 20 20 20 00	ld4 r21 = [r8] ↵
↳ // r21=k0	
00DC F0 09 2A 00	mov.i ar.lc = 31 ↵
↳ // TEA_ROUNDS is 32	
00E0 0A A0 00 06 10 10	ld4 r20 = [r3];; ↵
↳ // r20=k1	
00E6 20 01 80 20 20 00	ld4 r18 = [r32] ↵
↳ // r18=k3	
00EC 00 00 04 00	nop.i 0
00F0	loc_F0:

00F0 09 80 40 22 00 20	add r16 = r16, r17 ↗
↳ // r16=sum, r17=TEA_DELTA	
00F6 D0 71 54 26 40 80	shladd r29 = r14, 4, ↗
↳ r21 // r14=y, r21=k0	
00FC A3 70 68 52	extr.u r28 = r14, 5, ↗
↳ 27;;	
0100 03 F0 40 1C 00 20	add r30 = r16, r14
0106 B0 E1 50 00 40 40	add r27 = r28, r20;; ↗
↳ // r20=k1	
010C D3 F1 3C 80	xor r26 = r29, r30;;
0110 0B C8 6C 34 0F 20	xor r25 = r27, r26;;
0116 F0 78 64 00 40 00	add r15 = r15, r25 ↗
↳ // r15=z	
011C 00 00 04 00	nop.i 0;;
0120 00 00 00 00 01 00	nop.m 0
0126 80 51 3C 34 29 60	extr.u r24 = r15, 5, ↗
↳ 27	
012C F1 98 4C 80	shladd r11 = r15, 4, ↗
↳ r19 // r19=k2	
0130 0B B8 3C 20 00 20	add r23 = r15, r16;;
0136 A0 C0 48 00 40 00	add r10 = r24, r18 ↗
↳ // r18=k3	
013C 00 00 04 00	nop.i 0;;
0140 0B 48 28 16 0F 20	xor r9 = r10, r11;;
0146 60 B9 24 1E 40 00	xor r22 = r23, r9
014C 00 00 04 00	nop.i 0;;
0150 11 00 00 00 01 00	nop.m 0
0156 E0 70 58 00 40 A0	add r14 = r14, r22
015C A0 FF FF 48	br.cloop.sptk.few ↗
↳ loc_F0;;	
0160 09 20 3C 42 90 15	st4 [r33] = r15, 4 ↗
↳ // store z	
0166 00 00 00 02 00 00	nop.m 0
016C 20 08 AA 00	mov.i ar.lc = r2;; ↗
↳ // restore lc register	
0170 11 00 38 42 90 11	st4 [r33] = r14 ↗
↳ // store y	
0176 00 00 00 02 00 80	nop.i 0
017C 08 00 84 00	br.ret.sptk.many b0 ↗
↳ ;;	

. . .

.

. [b0-cc].

We also see a stop bit at 10c.

:

<http://go.yurichev.com/17322>.

: [Bur], [haq].

:

[MSDN](#).

Capítulo 94

([78.3 on page 1001](#) o [53.5 on page 783](#)), .

8086/8088 .

. 64KB. . . .

$$\text{real_address} = (\text{segment_register} \ll 4) + \text{address_register}$$

. 64KB.

Ejemplos ampliamente populares incluyen DOS/4GW (el videojuego DOOM fue compilado para él), Phar Lap, PMODE.

Capítulo 95

95.1 Profile-guided optimization

Listing 95.1: orageneric11.dll (win32)

```
public _skgfsync
proc near
; address 0x6030D86A

        db      66h
        nop
        push   ebp
        mov    ebp, esp
        mov    edx, [ebp+0Ch]
        test   edx, edx
        jz     short loc_6030D884
        mov    eax, [edx+30h]
        test   eax, 400h
        jnz    __VInfreq__skgfsync ; write to log
continue:
        mov    eax, [ebp+8]
        mov    edx, [ebp+10h]
        mov    dword ptr [eax], 0
        lea    eax, [edx+0Fh]
        and    eax, 0FFFFFFFCh
```

```

        mov    ecx, [eax]
        cmp    ecx, 45726963h
        jnz    error           ; exit with error
        mov    esp, ebp
        pop    ebp
        retn
_skgfsync endp

...
; address 0x60B953F0

_VInfreq_skgfsync:
        mov    eax, [edx]
        test   eax, eax
        jz     continue
        mov    ecx, [ebp+10h]
        push   ecx
        mov    ecx, [ebp+8]
        push   edx
        push   ecx
        push   offset ... ; "skgfsync(se=0x%x, ctx=0x%"
        ↵ x, iov=0x%x)\n"
        push   dword ptr [edx+4]
        call   dword ptr [eax] ; write to log
        add    esp, 14h
        jmp    continue

error:
        mov    edx, [ebp+8]
        mov    dword ptr [edx], 69AAh ; 27050 " ↵
        ↵ function called with invalid FIB/IOV structure"
        mov    eax, [eax]
        mov    [edx+4], eax
        mov    dword ptr [edx+8], 0FA4h ; 4004
        mov    esp, ebp
        pop    ebp
        retn
; END OF FUNCTION CHUNK FOR _skgfsync

```

Parte XI

Capítulo 96

96.1 Windows

[RA09].

96.2 C/C++

[ISO13].

96.3 x86 / x86-64

[Int13], [AMD13a]

96.4 ARM

: <http://go.yurichev.com/17024>

96.5

[Sch94]

Capítulo 97

97.1 Windows

- Microsoft: Raymond Chen
- nynaeve.net

Capítulo 98

: [reddit.com/r/ReverseEngineering/](https://www.reddit.com/r/ReverseEngineering/) y [reddit.com/r/remath/](https://www.reddit.com/r/remath/) ().

Stack Exchange :
reverseengineering.stackexchange.com.

##re FreeNode¹.

¹freenode.net

Parte XII

Ejercicios



1)

2) .

. .

Capítulo 99

1

99.1 Ejercicio 1.4

..

?

- win32 ([beginners.re](#))
- Linux x86 ([beginners.re](#))
- Mac OS X ([beginners.re](#))
- MIPS ([beginners.re](#))

Capítulo 100

2

100.1 Ejercicio 2.1

100.1.1 Con optimización MSVC 2010 x86

```
__real@3fe00000000000000 DQ 03fe00000000000000r
__real@3f50624dd2f1a9fc DQ 03f50624dd2f1a9fcr

_g$ = 8
tv132 = 16
_x$ = 16
f1 PROC
    fld     QWORD PTR _x$[esp-4]
    fld     QWORD PTR __real@3f50624dd2f1a9fc
    fld     QWORD PTR __real@3fe00000000000000
    fld     QWORD PTR _g$[esp-4]
$LN2@f1:
    fld     ST(0)
    fmul   ST(0), ST(1)
    fsub   ST(0), ST(4)
    call    __ftol2_sse
    cdq
    xor    eax, edx
    sub    eax, edx
    mov    DWORD PTR tv132[esp-4], eax
    fild   DWORD PTR tv132[esp-4]
    fcomp  ST(3)
    fnstsw ax
    test   ah, 5
    jnp    SHORT $LN19@f1
```

```

        fld    ST(3)
        fdiv   ST(0), ST(1)
        faddp  ST(1), ST(0)
        fmul   ST(0), ST(1)
        jmp    SHORT $LN2@f1
$LN19@f1:
        fstp   ST(3)
        fstp   ST(1)
        fstp   ST(0)
        ret    0
f1 ENDP

__real@3ff00000000000000 DQ 03ff0000000000000r

_x$ = 8
f2 PROC
        fld    QWORD PTR _x$[esp-4]
        sub    esp, 16
        fstp   QWORD PTR [esp+8]
        fld1
        fstp   QWORD PTR [esp]
        call   f1
        add    esp, 16
        ret    0
f2 ENDP

```

100.1.2 Con optimización MSVC 2012 x64

```

__real@3fe00000000000000 DQ 03fe0000000000000r
__real@3f50624dd2f1a9fc DQ 03f50624dd2f1a9fc
__real@3ff00000000000000 DQ 03ff0000000000000r

x$ = 8
f    PROC
        movsd xmm2, QWORD PTR __real@3ff00000000000000
        movsd xmm5, QWORD PTR __real@3f50624dd2f1a9fc
        movsd xmm4, QWORD PTR __real@3fe00000000000000
        movapd xmm3, xmm0
        npad    4
$LL4@f:
        movapd xmm1, xmm2
        mulsd  xmm1, xmm2
        subsd  xmm1, xmm3
        cvttsd2si eax, xmm1
        cdq
        xor    eax, edx

```

```

    sub    eax, edx
    movd   xmm0, eax
    cvtdq2pd xmm0, xmm0
    comisd xmm5, xmm0
    ja     SHORT $LN18@f
    movapd xmm0, xmm3
    divsd  xmm0, xmm2
    addsd  xmm0, xmm2
    movapd xmm2, xmm0
    mulsd  xmm2, xmm4
    jmp    SHORT $LL4@f
$LN18@f:
    movapd xmm0, xmm2
    ret    0
f      ENDP

```

100.2 Ejercicio 2.4

100.2.1 Con optimización MSVC 2010

```

PUBLIC _f
_TEXT SEGMENT
_arg1$ = 8           ; size = 4
_arg2$ = 12          ; size = 4
_f  PROC
    push  esi
    mov   esi, DWORD PTR _arg1$[esp]
    push  edi
    mov   edi, DWORD PTR _arg2$[esp+4]
    cmp   BYTE PTR [edi], 0
    mov   eax, esi
    je    SHORT $LN7@f
    mov   dl, BYTE PTR [esi]
    push  ebx
    test  dl, dl
    je    SHORT $LN4@f
    sub   esi, edi
    npad  6 ; align next label
$LL5@f:
    mov   ecx, edi
    test  dl, dl
    je    SHORT $LN2@f
$LL3@f:
    mov   dl, BYTE PTR [ecx]
    test  dl, dl
    je    SHORT $LN14@f

```

```

    movsx  ebx, BYTE PTR [esi+ecx]
    movsx  edx, dl
    sub    ebx, edx
    jne    SHORT $LN2@f
    inc    ecx
    cmp    BYTE PTR [esi+ecx], bl
    jne    SHORT $LL3@f
$LN2@f:
    cmp    BYTE PTR [ecx], 0
    je     SHORT $LN14@f
    mov    dl, BYTE PTR [eax+1]
    inc    eax
    inc    esi
    test   dl, dl
    jne    SHORT $LL5@f
    xor    eax, eax
    pop    ebx
    pop    edi
    pop    esi
    ret    0
_f      ENDP
_TEXT   ENDS
END

```

100.2.2 GCC 4.4.1

```

        public f
f          proc near

var_C      = dword ptr -0Ch
var_8      = dword ptr -8
var_4      = dword ptr -4
arg_0      = dword ptr  8
arg_4      = dword ptr  0Ch

        push   ebp
        mov    ebp, esp
        sub    esp, 10h
        mov    eax, [ebp+arg_0]
        mov    [ebp+var_4], eax
        mov    eax, [ebp+arg_4]
        movzx  eax, byte ptr [eax]
        test   al, al
        jnz   short loc_8048443
        mov    eax, [ebp+arg_0]
        jmp   short locret_8048453

```

```

loc_80483F4:
    mov     eax, [ebp+var_4]
    mov     [ebp+var_8], eax
    mov     eax, [ebp+arg_4]
    mov     [ebp+var_C], eax
    jmp     short loc_804840A

loc_8048402:
    add     [ebp+var_8], 1
    add     [ebp+var_C], 1

loc_804840A:
    mov     eax, [ebp+var_8]
    movzx  eax, byte ptr [eax]
    test   al, al
    jz     short loc_804842E
    mov     eax, [ebp+var_C]
    movzx  eax, byte ptr [eax]
    test   al, al
    jz     short loc_804842E
    mov     eax, [ebp+var_8]
    movzx  edx, byte ptr [eax]
    mov     eax, [ebp+var_C]
    movzx  eax, byte ptr [eax]
    cmp    dl, al
    jz     short loc_8048402

loc_804842E:
    mov     eax, [ebp+var_C]
    movzx  eax, byte ptr [eax]
    test   al, al
    jnz   short loc_804843D
    mov     eax, [ebp+var_4]
    jmp     short locret_8048453

loc_804843D:
    add     [ebp+var_4], 1
    jmp     short loc_8048444

loc_8048443:
    nop

loc_8048444:
    mov     eax, [ebp+var_4]
    movzx  eax, byte ptr [eax]

```

```

        test    al, al
        jnz     short loc_80483F4
        mov     eax, 0

locret_8048453:
        leave
        retn
f      endp

```

100.2.3 Con optimización Keil (Modo ARM)

```

PUSH    {r4,lr}
LDRB    r2,[r1,#0]
CMP     r2,#0
POPEQ   {r4,pc}
B       |L0.80|
|L0.20|
LDRB    r12,[r3,#0]
CMP     r12,#0
BEQ    |L0.64|
LDRB    r4,[r2,#0]
CMP     r4,#0
POPEQ   {r4,pc}
CMP     r12,r4
ADDEQ   r3,r3,#1
ADDEQ   r2,r2,#1
BEQ    |L0.20|
B       |L0.76|
|L0.64|
LDRB    r2,[r2,#0]
CMP     r2,#0
POPEQ   {r4,pc}
|L0.76|
ADD     r0,r0,#1
|L0.80|
LDRB    r2,[r0,#0]
CMP     r2,#0
MOVNE   r3,r0
MOVNE   r2,r1
MOVEQ   r0,#0
BNE    |L0.20|
POP     {r4,pc}

```

100.2.4 Con optimización Keil (Modo Thumb)

```

PUSH    {r4,r5,lr}
LDRB    r2,[r1,#0]
CMP     r2,#0
BEQ    |L0.54|
B      |L0.46|
|L0.10|
MOVS    r3,r0
MOVS    r2,r1
B      |L0.20|
|L0.16|
ADDS    r3,r3,#1
ADDS    r2,r2,#1
|L0.20|
LDRB    r4,[r3,#0]
CMP     r4,#0
BEQ    |L0.38|
LDRB    r5,[r2,#0]
CMP     r5,#0
BEQ    |L0.54|
CMP     r4,r5
BEQ    |L0.16|
B      |L0.44|
|L0.38|
LDRB    r2,[r2,#0]
CMP     r2,#0
BEQ    |L0.54|
|L0.44|
ADDS    r0,r0,#1
|L0.46|
LDRB    r2,[r0,#0]
CMP     r2,#0
BNE    |L0.10|
MOVS    r0,#0
|L0.54|
POP     {r4,r5,pc}

```

100.2.5 Con optimización GCC 4.9.1 (ARM64)

Listing 100.1: Con optimización GCC 4.9.1 (ARM64)

```

func:
    ldrb    w6, [x1]
    mov     x2, x0
    cbz    w6, .L2
    ldrb    w2, [x0]
    cbz    w2, .L24

```

```

.L17:
    ldrb    w2, [x0]
    cbz    w2, .L5
    cmp    w6, w2
    mov    x5, x0
    mov    x2, x1
    beq    .L18
    b     .L5

.L4:
    ldrb    w4, [x2]
    cmp    w3, w4
    cbz    w4, .L8
    bne    .L8

.L18:
    ldrb    w3, [x5,1]!
    add    x2, x2, 1
    cbnz   w3, .L4

.L8:
    ldrb    w2, [x2]
    cbz    w2, .L27

.L5:
    ldrb    w2, [x0,1]!
    cbnz   w2, .L17

.L24:
    mov    x2, 0

.L2:
    mov    x0, x2
    ret

.L27:
    mov    x2, x0
    mov    x0, x2
    ret

```

100.2.6 Con optimización GCC 4.4.5 (MIPS)

Listing 100.2: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:
    lb      $v1, 0($a1)
    or      $at, $zero
    bnez   $v1, loc_18
    move   $v0, $a0

locret_10:                                # CODE XREF: f+50
                                                # f+78
    jr      $ra
    or      $at, $zero

```

```

loc_18:                                # CODE XREF: f+8
    lb      $a0, 0($a0)
    or      $at, $zero
    beqz   $a0, locret_94
    move   $a2, $v0

loc_28:                                # CODE XREF: f+8C
    lb      $a0, 0($a2)
    or      $at, $zero
    beqz   $a0, loc_80
    or      $at, $zero
    bne    $v1, $a0, loc_80
    move   $a3, $a1
    b      loc_60
    addiu  $a2, 1

loc_48:                                # CODE XREF: f+68
    lb      $t1, 0($a3)
    or      $at, $zero
    beqz   $t1, locret_10
    or      $at, $zero
    bne    $t0, $t1, loc_80
    addiu  $a2, 1

loc_60:                                # CODE XREF: f+40
    lb      $t0, 0($a2)
    or      $at, $zero
    bnez   $t0, loc_48
    addiu  $a3, 1
    lb      $a0, 0($a3)
    or      $at, $zero
    beqz   $a0, locret_10
    or      $at, $zero

loc_80:                                # CODE XREF: f+30
                                         # f+38 ...
    addiu  $v0, 1
    lb      $a0, 0($v0)
    or      $at, $zero
    bnez   $a0, loc_28
    move   $a2, $v0

locret_94:                             # CODE XREF: f+20
    jr      $ra
    move   $v0, $zero

```

100.3 Ejercicio 2.6

100.3.1 Con optimización MSVC 2010

```

PUBLIC _f
; Function compile flags: /Ogtpy
_TEXT SEGMENT
_k0$ = -12           ; size = 4
_k3$ = -8            ; size = 4
_k2$ = -4            ; size = 4
_v$ = 8              ; size = 4
_k1$ = 12             ; size = 4
_k$ = 12              ; size = 4
_f PROC
    sub esp, 12      ; 0000000CH
    mov ecx, DWORD PTR _v$[esp+8]
    mov eax, DWORD PTR [ecx]
    mov ecx, DWORD PTR [ecx+4]
    push ebx
    push esi
    mov esi, DWORD PTR _k$[esp+16]
    push edi
    mov edi, DWORD PTR [esi]
    mov DWORD PTR _k0$[esp+24], edi
    mov edi, DWORD PTR [esi+4]
    mov DWORD PTR _k1$[esp+20], edi
    mov edi, DWORD PTR [esi+8]
    mov esi, DWORD PTR [esi+12]
    xor edx, edx
    mov DWORD PTR _k2$[esp+24], edi
    mov DWORD PTR _k3$[esp+24], esi
    lea edi, DWORD PTR [edx+32]
$L18@f:
    mov esi, ecx
    shr esi, 5
    add esi, DWORD PTR _k1$[esp+20]
    mov ebx, ecx
    shl ebx, 4
    add ebx, DWORD PTR _k0$[esp+24]
    sub edx, 1640531527 ; 61c88647H
    xor esi, ebx
    lea ebx, DWORD PTR [edx+ecx]
    xor esi, ebx
    add eax, esi
    mov esi, eax
    shr esi, 5
    add esi, DWORD PTR _k3$[esp+24]

```

```

    mov    ebx, eax
    shl    ebx, 4
    add    ebx, DWORD PTR _k2$[esp+24]
    xor    esi, ebx
    lea    ebx, DWORD PTR [edx+eax]
    xor    esi, ebx
    add    ecx, esi
    dec    edi
    jne    SHORT $LL8@f
    mov    edx, DWORD PTR _v$[esp+20]
    pop    edi
    pop    esi
    mov    DWORD PTR [edx], eax
    mov    DWORD PTR [edx+4], ecx
    pop    ebx
    add    esp, 12           ; 00000000cH
    ret    0
_f      ENDP

```

100.3.2 Con optimización Keil (Modo ARM)

```

PUSH   {r4-r10,lr}
ADD    r5,r1,#8
LDM    r5,{r5,r7}
LDR    r2,[r0,#4]
LDR    r3,[r0,#0]
LDR    r4,|L0.116|
LDR    r6,[r1,#4]
LDR    r8,[r1,#0]
MOV    r12,#0
MOV    r1,r12
|L0.40|
ADD    r12,r12,r4
ADD    r9,r8,r2,LSL #4
ADD    r10,r2,r12
EOR    r9,r9,r10
ADD    r10,r6,r2,LSR #5
EOR    r9,r9,r10
ADD    r3,r3,r9
ADD    r9,r5,r3,LSL #4
ADD    r10,r3,r12
EOR    r9,r9,r10
ADD    r10,r7,r3,LSR #5
EOR    r9,r9,r10
ADD    r1,r1,#1
CMP    r1,#0x20

```

ADD	r2,r2,r9
STRCS	r2,[r0,#4]
STRCS	r3,[r0,#0]
BCC	L0.40
POP	{r4-r10,pc}

L0.116	
DCD	0x9e3779b9

100.3.3 Con optimización Keil (Modo Thumb)

PUSH	{r1-r7,lr}
LDR	r5, L0.84
LDR	r3,[r0,#0]
LDR	r2,[r0,#4]
STR	r5,[sp,#8]
MOVS	r6,r1
LDM	r6,{r6,r7}
LDR	r5,[r1,#8]
STR	r6,[sp,#4]
LDR	r6,[r1,#0xc]
MOVS	r4,#0
MOVS	r1,r4
MOV	lr,r5
MOV	r12,r6
STR	r7,[sp,#0]

L0.30	
LDR	r5,[sp,#8]
LSLS	r6,r2,#4
ADDS	r4,r4,r5
LDR	r5,[sp,#4]
LSRS	r7,r2,#5
ADDS	r5,r6,r5
ADDS	r6,r2,r4
EORS	r5,r5,r6
LDR	r6,[sp,#0]
ADDS	r1,r1,#1
ADDS	r6,r7,r6
EORS	r5,r5,r6
ADDS	r3,r5,r3
LSLS	r5,r3,#4
ADDS	r6,r3,r4
ADD	r5,r5,lr
EORS	r5,r5,r6
LSRS	r6,r3,#5
ADD	r6,r6,r12

```

EORS      r5,r5,r6
ADDS      r2,r5,r2
CMP       r1,#0x20
BCC       |L0.30|
STR       r3,[r0,#0]
STR       r2,[r0,#4]
POP       {r1-r7,pc}

```

L0.84	DCD 0x9e3779b9
-------	----------------

100.3.4 Con optimización GCC 4.9.1 (ARM64)

Listing 100.3: Con optimización GCC 4.9.1 (ARM64)

```

f:
    ldr    w3, [x0]
    mov    w4, 0
    ldr    w2, [x0,4]
    ldr    w10, [x1]
    ldr   w9, [x1,4]
    ldr   w8, [x1,8]
    ldr   w7, [x1,12]
.L2:
    mov    w5, 31161
    add    w6, w10, w2, lsl 4
    movk  w5, 0x9e37, lsl 16
    add    w1, w9, w2, lsr 5
    add    w4, w4, w5
    eor    w1, w6, w1
    add    w5, w2, w4
    mov    w6, 14112
    eor    w1, w1, w5
    movk  w6, 0xc6ef, lsl 16
    add    w3, w3, w1
    cmp    w4, w6
    add    w5, w3, w4
    add    w6, w8, w3, lsl 4
    add    w1, w7, w3, lsr 5
    eor    w1, w6, w1
    eor    w1, w1, w5
    add    w2, w2, w1
    bne   .L2
    str   w3, [x0]
    str   w2, [x0,4]
    ret

```

100.3.5 Con optimización GCC 4.4.5 (MIPS)

Listing 100.4: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:
    lui      $t2, 0x9E37
    lui      $t1, 0xC6EF
    lw       $v0, 0($a0)
    lw       $v1, 4($a0)
    lw       $t6, 0xC($a1)
    lw       $t5, 0($a1)
    lw       $t4, 4($a1)
    lw       $t3, 8($a1)
    li       $t2, 0x9E3779B9
    li       $t1, 0xC6EF3720
    move    $a1, $zero

loc_2C:                      # CODE XREF: f+6C
    addu   $a1, $t2
    sll    $a2, $v1, 4
    addu   $t0, $a1, $v1
    srl    $a3, $v1, 5
    addu   $a2, $t5
    addu   $a3, $t4
    xor    $a2, $t0, $a2
    xor    $a2, $a3
    addu   $v0, $a2
    sll    $a3, $v0, 4
    srl    $a2, $v0, 5
    addu   $a3, $t3
    addu   $a2, $t6
    xor    $a2, $a3, $a2
    addu   $a3, $v0, $a1
    xor    $a2, $a3
    bne   $a1, $t1, loc_2C
    addu   $v1, $a2
    sw     $v1, 4($a0)
    jr    $ra
    sw     $v0, 0($a0)

```

100.4 Ejercicio 2.13

.?

100.4.1 Con optimización MSVC 2012

```

_in$ = 8                                ; size ↴
    ↳ = 2
_f      PROC
    movzx  ecx, WORD PTR _in$[esp-4]
    lea     eax, DWORD PTR [ecx*4]
    xor    eax, ecx
    add    eax, eax
    xor    eax, ecx
    shl    eax, 2
    xor    eax, ecx
    and   eax, 32
    ↳ 00000020H
    shl    eax, 10
    ↳ 0000000AH
    shr    ecx, 1
    or     eax, ecx
    ret    0
_f      ENDP

```

100.4.2 Keil (Modo ARM)

```

f PROC
    EOR      r1,r0,r0,LSR #2
    EOR      r1,r1,r0,LSR #3
    EOR      r1,r1,r0,LSR #5
    AND      r1,r1,#1
    LSR      r0,r0,#1
    ORR      r0,r0,r1,LSL #15
    BX       lr
    ENDP

```

100.4.3 Keil (Modo Thumb)

```

f PROC
    LSRS    r1,r0,#2
    EORS    r1,r1,r0
    LSRS    r2,r0,#3
    EORS    r1,r1,r2
    LSRS    r2,r0,#5
    EORS    r1,r1,r2
    LSLS    r1,r1,#31
    LSRS    r0,r0,#1
    LSRS    r1,r1,#16
    ORRS    r0,r0,r1

```

```
BX      lr
ENDP
```

100.4.4 Con optimización GCC 4.9.1 (ARM64)

```
f:
    uxth    w1, w0
    lsr     w2, w1, 3
    lsr     w0, w1, 1
    eor     w2, w2, w1, lsr 2
    eor     w2, w1, w2
    eor     w1, w2, w1, lsr 5
    and    w1, w1, 1
    orr    w0, w0, w1, lsl 15
    ret
```

100.4.5 Con optimización GCC 4.4.5 (MIPS)

Listing 100.5: Con optimización GCC 4.4.5 (MIPS) (IDA)

```
f:
    andi   $a0, 0xFFFF
    srl    $v1, $a0, 2
    srl    $v0, $a0, 3
    xor    $v0, $v1, $v0
    xor    $v0, $a0, $v0
    srl    $v1, $a0, 5
    xor    $v0, $v1
    andi   $v0, 1
    srl    $a0, 1
    sll    $v0, 15
    jr    $ra
    or    $v0, $a0
```

100.5 Ejercicio 2.14

100.5.1 MSVC 2012

```
_rt$1 = -4                                     ; size 2
    ↴ = 4
_rt$2 = 8                                     ; size 2
    ↴ = 4
_x$ = 8                                       ; size 2
    ↴ = 4
```

```

_y$ = 12 ; size 2
    ↓ = 4
?f@@YAIIZ PROC ; f
    push    ecx
    push    esi
    mov     esi, DWORD PTR _x$[esp+4]
    test   esi, esi
    jne    SHORT $LN7@f
    mov     eax, DWORD PTR _y$[esp+4]
    pop    esi
    pop    ecx
    ret    0

$LN7@f:
    mov     edx, DWORD PTR _y$[esp+4]
    mov     eax, esi
    test   edx, edx
    je     SHORT $LN8@f
    or     eax, edx
    push   edi
    bsf    edi, eax
    bsf    eax, esi
    mov     ecx, eax
    mov     DWORD PTR _rt$1[esp+12], eax
    bsf    eax, edx
    shr    esi, cl
    mov     ecx, eax
    shr    edx, cl
    mov     DWORD PTR _rt$2[esp+8], eax
    cmp    esi, edx
    je     SHORT $LN22@f

$LN23@f:
    jbe    SHORT $LN2@f
    xor    esi, edx
    xor    edx, esi
    xor    esi, edx

$LN2@f:
    cmp    esi, 1
    je     SHORT $LN22@f
    sub    edx, esi
    bsf    eax, edx
    mov     ecx, eax
    shr    edx, cl
    mov     DWORD PTR _rt$2[esp+8], eax
    cmp    esi, edx
    jne    SHORT $LN23@f

$LN22@f:
    mov     ecx, edi

```

```

    shl    esi, cl
    pop    edi
    mov    eax, esi
$LN8@f:
    pop    esi
    pop    ecx
    ret    0
?f@@YAIII@Z ENDP

```

100.5.2 Keil (Modo ARM)

```

||f1|| PROC
    CMP    r0,#0
    RSB    r1,r0,#0
    AND    r0,r0,r1
    CLZ    r0,r0
    RSBNE r0,r0,#0x1f
    BX    lr
    ENDP

```

```

f PROC
    MOVS   r2,r0
    MOV    r3,r1
    MOVEQ  r0,r1
    CMPNE r3,#0
    PUSH   {lr}
    POPEQ  {pc}
    ORR    r0,r2,r3
    BL    ||f1||
    MOV    r12,r0
    MOV    r0,r2
    BL    ||f1||
    LSR    r2,r2,r0

```

```

|L0.196|
    MOV    r0,r3
    BL    ||f1||
    LSR    r0,r3,r0
    CMP    r2,r0
    EORHI r1,r2,r0
    EORHI r0,r0,r1
    EORHI r2,r1,r0
    BEQ   |L0.240|
    CMP    r2,#1
    SUBNE r3,r0,r2
    BNE   |L0.196|

```

```
|L0.240|
```

LSL	r0, r2, r12
POP	{pc}
ENDP	

100.5.3 GCC 4.6.3 for Raspberry Pi (Modo ARM)

```
f:
    subs    r3, r0, #0
    beq     .L162
    cmp     r1, #0
    moveq   r1, r3
    beq     .L162
    orr     r2, r1, r3
    rsb     ip, r2, #0
    and    ip, ip, r2
    cmp     r2, #0
    rsb     r2, r3, #0
    and    r2, r2, r3
    clz     r2, r2
    rsb     r2, r2, #31
    clz     ip, ip
    rsbne  ip, ip, #31
    mov     r3, r3, lsr r2
    b      .L169

.L171:
    eorhi   r1, r1, r2
    eorhi   r3, r1, r2
    cmp     r3, #1
    rsb     r1, r3, r1
    beq     .L167

.L169:
    rsb     r0, r1, #0
    and    r0, r0, r1
    cmp     r1, #0
    clz     r0, r0
    mov     r2, r0
    rsbne  r2, r0, #31
    mov     r1, r1, lsr r2
    cmp     r3, r1
    eor     r2, r1, r3
    bne     .L171

.L167:
    mov     r1, r3, asl ip

.L162:
    mov     r0, r1
    bx      lr
```

100.5.4 Con optimización GCC 4.9.1 (ARM64)

Listing 100.6: Con optimización GCC 4.9.1 (ARM64)

```

f:
    mov      w3, w0
    mov      w0, w1
    cbz     w3, .L8
    mov      w0, w3
    cbz     w1, .L8
    mov      w6, 31
    orr     w5, w3, w1
    neg      w2, w3
    neg      w7, w5
    and     w2, w2, w3
    clz      w2, w2
    sub     w2, w6, w2
    and     w5, w7, w5
    mov      w4, w6
    clz      w5, w5
    lsr     w0, w3, w2
    sub     w5, w6, w5
    b       .L13

.L22:
    bls     .L12
    eor     w1, w1, w2
    eor     w0, w1, w2

.L12:
    cmp     w0, 1
    sub     w1, w1, w0
    beq     .L11

.L13:
    neg      w2, w1
    cmp     w1, wzr
    and     w2, w2, w1
    clz      w2, w2
    sub     w3, w4, w2
    csel   w2, w3, w2, ne
    lsr     w1, w1, w2
    cmp     w0, w1
    eor     w2, w1, w0
    bne     .L22

.L11:
    lsl     w0, w0, w5

.L8:
    ret

```

100.5.5 Con optimización GCC 4.4.5 (MIPS)

Listing 100.7: Con optimización GCC 4.4.5 (MIPS) (IDA)

```

f:

var_20      = -0x20
var_18      = -0x18
var_14      = -0x14
var_10      = -0x10
var_C       = -0xC
var_8       = -8
var_4       = -4

lui      $gp, (__gnu_local_gp >> 16)
addiu   $sp, -0x30
la       $gp, (__gnu_local_gp & 0xFFFF)
sw       $ra, 0x30+var_4($sp)
sw       $s4, 0x30+var_8($sp)
sw       $s3, 0x30+var_C($sp)
sw       $s2, 0x30+var_10($sp)
sw       $s1, 0x30+var_14($sp)
sw       $s0, 0x30+var_18($sp)
sw       $gp, 0x30+var_20($sp)
move    $s0, $a0
beqz   $a0, loc_154
move    $s1, $a1
bnez   $a1, loc_178
or      $s2, $a1, $a0
move    $s1, $a0

loc_154:          # CODE XREF: f+2C
lw      $ra, 0x30+var_4($sp)
move   $v0, $s1
lw      $s4, 0x30+var_8($sp)
lw      $s3, 0x30+var_C($sp)
lw      $s2, 0x30+var_10($sp)
lw      $s1, 0x30+var_14($sp)
lw      $s0, 0x30+var_18($sp)
jr      $ra
addiu $sp, 0x30

loc_178:          # CODE XREF: f+34
lw      $t9, (__clzsi2 & 0xFFFF)($gp)
negu   $a0, $s2
jalr   $t9
and    $a0, $s2
lw      $gp, 0x30+var_20($sp)

```

```

        bnez    $s2, loc_20C
        li      $s4, 0x1F
        move   $s4, $v0

loc_198:                                # CODE XREF: f:loc_20C
        lw      $t9, (__clzsi2 & 0xFFFF)($gp)
        negu   $a0, $s0
        jalr   $t9
        and    $a0, $s0
        nor    $v0, $zero, $v0
        lw      $gp, 0x30+var_20($sp)
        srlv   $s0, $v0
        li      $s3, 0x1F
        li      $s2, 1

loc_1BC:                                # CODE XREF: f+F0
        lw      $t9, (__clzsi2 & 0xFFFF)($gp)
        negu   $a0, $s1
        jalr   $t9
        and    $a0, $s1
        lw      $gp, 0x30+var_20($sp)
        beqz   $s1, loc_1DC
        or     $at, $zero
        subu   $v0, $s3, $v0

loc_1DC:                                # CODE XREF: f+BC
        srlv   $s1, $v0
        xor    $v1, $s1, $s0
        beq    $s0, $s1, loc_214
        sltu   $v0, $s1, $s0
        beqz   $v0, loc_1FC
        or     $at, $zero
        xor    $s1, $v1
        xor    $s0, $s1, $v1

loc_1FC:                                # CODE XREF: f+D8
        beq    $s0, $s2, loc_214
        subu   $s1, $s0
        b      loc_1BC
        or     $at, $zero

loc_20C:                                # CODE XREF: f+78
        b      loc_198
        subu   $s4, $v0

loc_214:                                # CODE XREF: f+D0
                                            # f:loc_1FC

```

```

lw      $ra, 0x30+var_4($sp)
sllv   $s1, $s0, $s4
move   $v0, $s1
lw      $s4, 0x30+var_8($sp)
lw      $s3, 0x30+var_C($sp)
lw      $s2, 0x30+var_10($sp)
lw      $s1, 0x30+var_14($sp)
lw      $s0, 0x30+var_18($sp)
jr      $ra
addiu  $sp, 0x30

```

100.6 Ejercicio 2.15

[27 on page 560.](#)

100.6.1 Con optimización MSVC 2012 x64

```

__real@412e848000000000 DQ 0412e84800000000000r ; 1e+006
__real@4010000000000000 DQ 0401000000000000000r ; 4
__real@4008000000000000 DQ 0400800000000000000r ; 3
__real@3f800000 DD 03f800000r ; 1

tmp$1 = 8
tmp$2 = 8
f      PROC
        movsd xmm3, QWORD PTR __real@4008000000000000
        movss xmm4, DWORD PTR __real@3f800000
        mov    edx, DWORD PTR ?RNG_state@?1??get_rand@@9@9
        xor    ecx, ecx
        mov    r8d, 200000 ; 00030✓
    ↳ d40H
        npad   2 ; align next label
$LL4@f:
        imul   edx, 1664525 ; ✓
    ↳ 0019660dH
        add    edx, 1013904223 ; 3✓
    ↳ c6ef35fH
        mov    eax, edx
        and    eax, 8388607 ; 007✓
    ↳ fffffH
        imul   edx, 1664525 ; ✓
    ↳ 0019660dH
        bts    eax, 30
        add    edx, 1013904223 ; 3✓
    ↳ c6ef35fH

```

```

    mov     DWORD PTR tmp$2[rsp], eax
    mov     eax, edx
    and     eax, 8388607 ; 007✓
    ↳ fffffH
        bts     eax, 30
        movss   xmm0, DWORD PTR tmp$2[rsp]
        mov     DWORD PTR tmp$1[rsp], eax
        cvtps2pd xmm0, xmm0
        subsd   xmm0, xmm3
        cvtpd2ps xmm2, xmm0
        movss   xmm0, DWORD PTR tmp$1[rsp]
        cvtps2pd xmm0, xmm0
        mulss   xmm2, xmm2
        subsd   xmm0, xmm3
        cvtpd2ps xmm1, xmm0
        mulss   xmm1, xmm1
        addss   xmm1, xmm2
        comiss  xmm4, xmm1
        jbe     SHORT $LN3@f
        inc     ecx

$LN3@f:
    imul   edx, 1664525 ; ✓
    ↳ 0019660dH
        add     edx, 1013904223 ; 3✓
    ↳ c6ef35fH
        mov     eax, edx
        and     eax, 8388607 ; 007✓
    ↳ fffffH
        imul   edx, 1664525 ; ✓
    ↳ 0019660dH
        bts     eax, 30
        add     edx, 1013904223 ; 3✓
    ↳ c6ef35fH
        mov     DWORD PTR tmp$2[rsp], eax
        mov     eax, edx
        and     eax, 8388607 ; 007✓
    ↳ fffffH
        bts     eax, 30
        movss   xmm0, DWORD PTR tmp$2[rsp]
        mov     DWORD PTR tmp$1[rsp], eax
        cvtps2pd xmm0, xmm0
        subsd   xmm0, xmm3
        cvtpd2ps xmm2, xmm0
        movss   xmm0, DWORD PTR tmp$1[rsp]
        cvtps2pd xmm0, xmm0
        mulss   xmm2, xmm2
        subsd   xmm0, xmm3

```

```

        cvtpd2ps xmm1, xmm0
        mulss  xmm1, xmm1
        addss  xmm1, xmm2
        comiss xmm4, xmm1
        jbe    SHORT $LN15@f
        inc    ecx
$LN15@f:
        imul   edx, 1664525 ; ↴
        ↳ 0019660dH
            add    edx, 1013904223 ; 3 ↴
        ↳ c6ef35fH
            mov    eax, edx
            and    eax, 8388607 ; 007 ↴
        ↳ fffffH
            imul   edx, 1664525 ; ↴
        ↳ 0019660dH
            bts   eax, 30
            add    edx, 1013904223 ; 3 ↴
        ↳ c6ef35fH
            mov    DWORD PTR tmp$2[rsp], eax
            mov    eax, edx
            and    eax, 8388607 ; 007 ↴
        ↳ fffffH
            bts   eax, 30
            movss  xmm0, DWORD PTR tmp$2[rsp]
            mov    DWORD PTR tmp$1[rsp], eax
            cvtps2pd xmm0, xmm0
            subsd  xmm0, xmm3
            cvtpd2ps xmm2, xmm0
            movss  xmm0, DWORD PTR tmp$1[rsp]
            cvtps2pd xmm0, xmm0
            mulss  xmm2, xmm2
            subsd  xmm0, xmm3
            cvtpd2ps xmm1, xmm0
            mulss  xmm1, xmm1
            addss  xmm1, xmm2
            comiss xmm4, xmm1
            jbe    SHORT $LN16@f
            inc    ecx
$LN16@f:
        imul   edx, 1664525 ; ↴
        ↳ 0019660dH
            add    edx, 1013904223 ; 3 ↴
        ↳ c6ef35fH
            mov    eax, edx
            and    eax, 8388607 ; 007 ↴
        ↳ fffffH

```

```

        imul    edx, 1664525 ; ↵
↳ 0019660dH
        bts     eax, 30
        add     edx, 1013904223 ; 3 ↵
↳ c6ef35fH
        mov     DWORD PTR tmp$2[rsp], eax
        mov     eax, edx
        and     eax, 8388607 ; 007 ↵
↳ fffffH
        bts     eax, 30
        movss  xmm0, DWORD PTR tmp$2[rsp]
        mov     DWORD PTR tmp$1[rsp], eax
        cvtps2pd xmm0, xmm0
        subsd  xmm0, xmm3
        cvtpd2ps xmm2, xmm0
        movss  xmm0, DWORD PTR tmp$1[rsp]
        cvtps2pd xmm0, xmm0
        mulss  xmm2, xmm2
        subsd  xmm0, xmm3
        cvtpd2ps xmm1, xmm0
        mulss  xmm1, xmm1
        addss  xmm1, xmm2
        comiss xmm4, xmm1
        jbe    SHORT $LN17@f
        inc    ecx

$LN17@f:
        imul    edx, 1664525 ; ↵
↳ 0019660dH
        add     edx, 1013904223 ; 3 ↵
↳ c6ef35fH
        mov     eax, edx
        and     eax, 8388607 ; 007 ↵
↳ fffffH
        imul    edx, 1664525 ; ↵
↳ 0019660dH
        bts     eax, 30
        add     edx, 1013904223 ; 3 ↵
↳ c6ef35fH
        mov     DWORD PTR tmp$2[rsp], eax
        mov     eax, edx
        and     eax, 8388607 ; 007 ↵
↳ fffffH
        bts     eax, 30
        movss  xmm0, DWORD PTR tmp$2[rsp]
        mov     DWORD PTR tmp$1[rsp], eax
        cvtps2pd xmm0, xmm0
        subsd  xmm0, xmm3

```

```

cvt生产力2ps xmm2, xmm0
movss  xmm0, DWORD PTR tmp$1[rsp]
cvt生产力2pd xmm0, xmm0
mulss  xmm2, xmm2
subsd  xmm0, xmm3
cvt生产力2ps xmm1, xmm0
mulss  xmm1, xmm1
addss  xmm1, xmm2
comiss xmm4, xmm1
jbe    SHORT $LN18@f
inc    ecx

$LN18@f:
dec    r8
jne    $LL4@f
movd  xmm0, ecx
mov   DWORD PTR ?RNG_state@?1??get_rand@@9@9, edx
cvtdq2ps xmm0, xmm0
cvt生产力2pd xmm1, xmm0
mulsd xmm1, QWORD PTR __real@4010000000000000
divsd xmm1, QWORD PTR __real@412e848000000000
cvt生产力2ps xmm0, xmm1
ret    0
f      ENDP

```

100.6.2 Con optimización GCC 4.4.6 x64

```

f1:
    mov    eax, DWORD PTR v1.2084[rip]
    imul   eax, eax, 1664525
    add    eax, 1013904223
    mov    DWORD PTR v1.2084[rip], eax
    and    eax, 8388607
    or     eax, 1073741824
    mov    DWORD PTR [rsp-4], eax
    movss  xmm0, DWORD PTR [rsp-4]
    subss  xmm0, DWORD PTR .LC0[rip]
    ret

f:
    push   rbp
    xor    ebp, ebp
    push   rbx
    xor    ebx, ebx
    sub    rsp, 16

.L6:
    xor    eax, eax
    call   f1

```

```

xor    eax, eax
movss DWORD PTR [rsp], xmm0
call   f1
movss xmm1, DWORD PTR [rsp]
mulss xmm0, xmm0
mulss xmm1, xmm1
lea    eax, [rbx+1]
addss xmm1, xmm0
movss xmm0, DWORD PTR .LC1[rip]
ucomiss xmm0, xmm1
cmova ebx, eax
add    ebp, 1
cmp    ebp, 1000000
jne    .L6
cvtsi2ss      xmm0, ebx
unpcklps     xmm0, xmm0
cvtps2pd     xmm0, xmm0
mulsd  xmm0, QWORD PTR .LC2[rip]
divsd  xmm0, QWORD PTR .LC3[rip]
add    rsp, 16
pop    rbx
pop    rbp
unpcklpd    xmm0, xmm0
cvtpd2ps    xmm0, xmm0
ret
v1.2084:
.long   305419896
.LC0:
.long   1077936128
.LC1:
.long   1065353216
.LC2:
.long   0
.long   1074790400
.LC3:
.long   0
.long   1093567616

```

100.6.3 Con optimización GCC 4.8.1 x86

```

f1:
sub    esp, 4
imul  eax, DWORD PTR v1.2023, 1664525
add    eax, 1013904223
mov    DWORD PTR v1.2023, eax
and    eax, 8388607

```

```

        or      eax, 1073741824
        mov     DWORD PTR [esp], eax
        fld     DWORD PTR [esp]
        fsub   DWORD PTR .LC0
        add    esp, 4
        ret

f:
        push   esi
        mov    esi, 1000000
        push   ebx
        xor    ebx, ebx
        sub    esp, 16

.L7:
        call   f1
        fstp  DWORD PTR [esp]
        call   f1
        lea    eax, [ebx+1]
        fld    DWORD PTR [esp]
        fmul  st, st(0)
        fxch  st(1)
        fmul  st, st(0)
        faddp st(1), st
        fld1
        fucomip st, st(1)
        fstp  st(0)
        cmova ebx, eax
        sub    esi, 1
        jne   .L7
        mov    DWORD PTR [esp+4], ebx
        fild  DWORD PTR [esp+4]
        fmul  DWORD PTR .LC3
        fdiv  DWORD PTR .LC4
        fstp  DWORD PTR [esp+8]
        fld   DWORD PTR [esp+8]
        add    esp, 16
        pop    ebx
        pop    esi
        ret

v1.2023:
        .long  305419896
.LC0:
        .long  1077936128
.LC3:
        .long  1082130432
.LC4:
        .long  1232348160

```

100.6.4 Keil (Modo ARM):

```

f1      PROC
        LDR      r1, |L0.184|
        LDR      r0,[r1,#0] ; v1
        LDR      r2,|L0.188|
        VMOV.F32 s1,#3.00000000
        MUL      r0,r0,r2
        LDR      r2,|L0.192|
        ADD      r0,r0,r2
        STR      r0,[r1,#0] ; v1
        BFC      r0,#23,#9
        ORR      r0,r0,#0x40000000
        VMOV    s0,r0
        VSUB.F32 s0,s0,s1
        BX      lr
        ENDP

f      PROC
        PUSH     {r4,r5,lr}
        MOV      r4,#0
        LDR      r5,|L0.196|
        MOV      r3,r4
|L0.68|
        BL      f1
        VMOV.F32 s2,s0
        BL      f1
        VMOV.F32 s1,s2
        ADD      r3,r3,#1
        VMUL.F32 s1,s1,s1
        VMLA.F32 s1,s0,s0
        VMOV    r0,s1
        CMP      r0,#0x3f800000
        ADDLT   r4,r4,#1
        CMP      r3,r5
        BLT     |L0.68|
        VMOV    s0,r4
        VMOV.F64 d1,#4.00000000
        VCVT.F32.S32 s0,s0
        VCVT.F64.F32 d0,s0
        VMUL.F64 d0,d0,d1
        VLDR    d1,|L0.200|
        VDIV.F64 d2,d0,d1
        VCVT.F32.F64 s0,d2
        POP      {r4,r5,pc}
        ENDP

|L0.184|

```

```

    DCD    |||.data||
|L0.188|
    DCD    0x0019660d
|L0.192|
    DCD    0x3c6ef35f
|L0.196|
    DCD    0x000f4240
|L0.200|
    DCFD   0x412e848000000000 ; 1000000
    DCD    0x00000000
    AREA  |||.data||, DATA, ALIGN=2
v1
    DCD    0x12345678

```

100.6.5 Con optimización GCC 4.9.1 (ARM64)

Listing 100.8: Con optimización GCC 4.9.1 (ARM64)

```

f1:
    adrp    x2, .LANCHOR0
    mov     w3, 26125
    mov     w0, 62303
    movk   w3, 0x19, lsl 16
    movk   w0, 0x3c6e, lsl 16
    ldr    w1, [x2,#:lo12:.LANCHOR0]
    fmov   s0, 3.0e+0
    madd   w0, w1, w3, w0
    str    w0, [x2,#:lo12:.LANCHOR0]
    and    w0, w0, 8388607
    orr    w0, w0, 1073741824
    fmov   s1, w0
    fsub   s0, s1, s0
    ret
main_function:
    adrp    x7, .LANCHOR0
    mov     w3, 16960
    movk   w3, 0xf, lsl 16
    mov     w2, 0
    fmov   s2, 3.0e+0
    ldr    w1, [x7,#:lo12:.LANCHOR0]
    fmov   s3, 1.0e+0
.L5:
    mov     w6, 26125
    mov     w0, 62303
    movk   w6, 0x19, lsl 16
    movk   w0, 0x3c6e, lsl 16

```

```

    mov    w5, 26125
    mov    w4, 62303
    madd   w1, w1, w6, w0
    movk   w5, 0x19, lsl 16
    movk   w4, 0x3c6e, lsl 16
    and    w0, w1, 8388607
    add    w6, w2, 1
    orr    w0, w0, 1073741824
    madd   w1, w1, w5, w4
    fmov   s0, w0
    and    w0, w1, 8388607
    orr    w0, w0, 1073741824
    fmov   s1, w0
    fsub   s0, s0, s2
    fsub   s1, s1, s2
    fmul   s1, s1, s1
    fmadd  s0, s0, s0, s1
    fcmp   s0, s3
    csel   w2, w2, w6, p1
    subs   w3, w3, #1
    bne    .L5
    scvtf  s0, w2
    str    w1, [x7,#:lo12:.LANCHOR0]
    fmov   d1, 4.0e+0
    fcvt   d0, s0
    fmul   d0, d0, d1
    ldr    d1, .LC0
    fdiv   d0, d0, d1
    fcvt   s0, d0
    ret

.LC0:
    .word  0
    .word  1093567616
v1:
    .word  1013904223
v2:
    .word  1664525
.LANCHOR0 = . + 0
v3.3095:
    .word  305419896

```

100.6.6 Con optimización GCC 4.4.5 (MIPS)

Listing 100.9: Con optimización GCC 4.4.5 (MIPS) (IDA)

f1:	mov eax, DWORD PTR v1.2084[rip]
-----	---------------------------------

```

imul    eax, eax, 1664525
add     eax, 1013904223
mov     DWORD PTR v1.2084[rip], eax
and     eax, 8388607
or      eax, 1073741824
mov     DWORD PTR [rsp-4], eax
movss   xmm0, DWORD PTR [rsp-4]
subss   xmm0, DWORD PTR .LC0[rip]
ret

f:
push    rbp
xor    ebp, ebp
push    rbx
xor    ebx, ebx
sub    rsp, 16

.L6:
xor    eax, eax
call   f1
xor    eax, eax
movss   DWORD PTR [rsp], xmm0
call   f1
movss   xmm1, DWORD PTR [rsp]
mulss   xmm0, xmm0
mulss   xmm1, xmm1
lea     eax, [rbx+1]
addss   xmm1, xmm0
movss   xmm0, DWORD PTR .LC1[rip]
ucomiss xmm0, xmm1
cmova  ebx, eax
add    ebp, 1
cmp    ebp, 1000000
jne    .L6
cvtsi2ss    xmm0, ebx
unpcklps   xmm0, xmm0
cvtpd2pd   xmm0, xmm0
mulsd    xmm0, QWORD PTR .LC2[rip]
divsd    xmm0, QWORD PTR .LC3[rip]
add     rsp, 16
pop     rbx
pop     rbp
unpcklpd   xmm0, xmm0
cvtpd2ps   xmm0, xmm0
ret

v1.2084:
.long   305419896
.LC0:
.long   1077936128

```

```
.LC1:
    .long    1065353216
.LC2:
    .long    0
    .long    1074790400
.LC3:
    .long    0
    .long    1093567616
```

100.7 Ejercicio 2.16

100.7.1 Con optimización MSVC 2012 x64

```
m$ = 48
n$ = 56
f PROC
$LN14:
    push    rbx
    sub     rsp, 32
    mov     eax, edx
    mov     ebx, ecx
    test    ecx, ecx
    je      SHORT $LN11@f
$LL5@f:
    test    eax, eax
    jne     SHORT $LN1@f
    mov     eax, 1
    jmp     SHORT $LN12@f
$LN1@f:
    lea     edx, DWORD PTR [rax-1]
    mov     ecx, ebx
    call    f
$LN12@f:
    dec     ebx
    test    ebx, ebx
    jne     SHORT $LL5@f
$LN11@f:
    inc     eax
    add     rsp, 32
    pop     rbx
    ret     0
f ENDP
```

100.7.2 Con optimización Keil (Modo ARM)

```
f PROC
    PUSH    {r4,lr}
    MOVS    r4,r0
    ADDEQ   r0,r1,#1
    POPEQ   {r4,pc}
    CMP     r1,#0
    MOVEQ   r1,#1
    SUBEQ   r0,r0,#1
    BEQ    |L0.48|
    SUB    r1,r1,#1
    BL     f
    MOV    r1,r0
    SUB    r0,r4,#1
|L0.48|
    POP    {r4,lr}
    B      f
    ENDP
```

100.7.3 Con optimización Keil (Modo Thumb)

```
f PROC
    PUSH    {r4,lr}
    MOVS    r4,r0
    BEQ    |L0.26|
    CMP    r1,#0
    BEQ    |L0.30|
    SUBS   r1,r1,#1
    BL     f
    MOVS   r1,r0
|L0.18|
    SUBS   r0,r4,#1
    BL     f
    POP    {r4,pc}
|L0.26|
    ADDS   r0,r1,#1
    POP    {r4,pc}
|L0.30|
    MOVS   r1,#1
    B      |L0.18|
    ENDP
```

100.7.4 Sin optimización GCC 4.9.1 (ARM64)

Listing 100.10: Sin optimización GCC 4.9.1 (ARM64)

```

f:
    stp    x29, x30, [sp, -48]!
    add    x29, sp, 0
    str    x19, [sp,16]
    str    w0, [x29,44]
    str    w1, [x29,40]
    ldr    w0, [x29,44]
    cmp    w0, wzr
    bne   .L2
    ldr    w0, [x29,40]
    add    w0, w0, 1
    b     .L3

.L2:
    ldr    w0, [x29,40]
    cmp    w0, wzr
    bne   .L4
    ldr    w0, [x29,44]
    sub   w0, w0, #1
    mov    w1, 1
    bl    ack
    b     .L3

.L4:
    ldr    w0, [x29,44]
    sub   w19, w0, #1
    ldr    w0, [x29,40]
    sub   w1, w0, #1
    ldr    w0, [x29,44]
    bl    ack
    mov    w1, w0
    mov    w0, w19
    bl    ack

.L3:
    ldr    x19, [sp,16]
    ldp    x29, x30, [sp], 48
    ret

```

100.7.5 Con optimización GCC 4.9.1 (ARM64)

Listing 100.11: Con optimización GCC 4.9.1 (ARM64)

```

ack:
    stp    x29, x30, [sp, -160]!
    add    x29, sp, 0
    stp    d8, d9, [sp,96]
    stp    x19, x20, [sp,16]
    stp    d10, d11, [sp,112]
    stp    x21, x22, [sp,32]

```

```

    stp    d12, d13, [sp,128]
    stp    x23, x24, [sp,48]
    stp    d14, d15, [sp,144]
    stp    x25, x26, [sp,64]
    stp    x27, x28, [sp,80]
    cbz    w0, .L2
    sub    w0, w0, #1
    fmov   s10, w0
    b     .L4

.L46:
    fmov   w0, s10
    mov    w1, 1
    sub    w0, w0, #1
    fmov   s10, w0
    fmov   w0, s13
    cbz    w0, .L2

.L4:
    fmov   s13, s10
    cbz    w1, .L46
    sub    w1, w1, #1
    fmov   s11, s10
    b     .L7

.L48:
    fmov   w0, s11
    mov    w1, 1
    sub    w0, w0, #1
    fmov   s11, w0
    fmov   w0, s14
    cbz    w0, .L47

.L7:
    fmov   s14, s11
    cbz    w1, .L48
    sub    w1, w1, #1
    fmov   s12, s11
    b     .L10

.L50:
    fmov   w0, s12
    mov    w1, 1
    sub    w0, w0, #1
    fmov   s12, w0
    fmov   w0, s15
    cbz    w0, .L49

.L10:
    fmov   s15, s12
    cbz    w1, .L50
    sub    w1, w1, #1
    fmov   s8, s12

```

```

        b      .L13
.L52:
    fmov   w0, s8
    mov    w1, 1
    sub    w0, w0, #1
    fmov   s8, w0
    fmov   w0, s9
    cbz   w0, .L51
.L13:
    fmov   s9, s8
    cbz   w1, .L52
    sub    w1, w1, #1
    fmov   w22, s8
    b     .L16
.L54:
    mov    w1, 1
    sub    w22, w22, #1
    cbz   w28, .L53
.L16:
    mov    w28, w22
    cbz   w1, .L54
    sub    w1, w1, #1
    mov    w21, w22
    b     .L19
.L56:
    mov    w1, 1
    sub    w21, w21, #1
    cbz   w24, .L55
.L19:
    mov    w24, w21
    cbz   w1, .L56
    sub    w1, w1, #1
    mov    w20, w21
    b     .L22
.L58:
    mov    w1, 1
    sub    w20, w20, #1
    cbz   w25, .L57
.L22:
    mov    w25, w20
    cbz   w1, .L58
    sub    w1, w1, #1
    mov    w26, w20
    b     .L25
.L60:
    mov    w1, 1
    sub    w26, w26, #1

```

```

        cbz    w27, .L59
.L25:
        mov    w27, w26
        cbz    w1, .L60
        sub    w1, w1, #1
        mov    w19, w26
        b     .L28
.L62:
        mov    w23, w19
        mov    w1, 1
        sub    w19, w19, #1
        cbz    w23, .L61
.L28:
        add    w0, w19, 1
        cbz    w1, .L62
        sub    w1, w1, #1
        mov    w23, w19
        sub    w19, w19, #1
        bl     ack
        mov    w1, w0
        cbnz   w23, .L28
.L61:
        add    w1, w1, 1
        sub    w26, w26, #1
        cbnz   w27, .L25
.L59:
        add    w1, w1, 1
        sub    w20, w20, #1
        cbnz   w25, .L22
.L57:
        add    w1, w1, 1
        sub    w21, w21, #1
        cbnz   w24, .L19
.L55:
        add    w1, w1, 1
        sub    w22, w22, #1
        cbnz   w28, .L16
.L53:
        fmov   w0, s8
        add    w1, w1, 1
        sub    w0, w0, #1
        fmov   s8, w0
        fmov   w0, s9
        cbnz   w0, .L13
.L51:
        fmov   w0, s12
        add    w1, w1, 1

```

```

    sub    w0, w0, #1
    fmov  s12, w0
    fmov  w0, s15
    cbnz  w0, .L10

.L49:
    fmov  w0, s11
    add   w1, w1, 1
    sub   w0, w0, #1
    fmov  s11, w0
    fmov  w0, s14
    cbnz  w0, .L7

.L47:
    fmov  w0, s10
    add   w1, w1, 1
    sub   w0, w0, #1
    fmov  s10, w0
    fmov  w0, s13
    cbnz  w0, .L4

.L2:
    add   w0, w1, 1
    ldp   d8, d9, [sp,96]
    ldp   x19, x20, [sp,16]
    ldp   d10, d11, [sp,112]
    ldp   x21, x22, [sp,32]
    ldp   d12, d13, [sp,128]
    ldp   x23, x24, [sp,48]
    ldp   d14, d15, [sp,144]
    ldp   x25, x26, [sp,64]
    ldp   x27, x28, [sp,80]
    ldp   x29, x30, [sp], 160
    ret

```

100.7.6 Sin optimización GCC 4.4.5 (MIPS)

Listing 100.12: Sin optimización GCC 4.4.5 (MIPS) (IDA)

f:	# CODE XREF: f+64 # f+94 ...
var_C	= -0xC
var_8	= -8
var_4	= -4
arg_0	= 0
arg_4	= 4
	addiu \$sp, -0x28
	sw \$ra, 0x28+var_4(\$sp)

```

        sw      $fp, 0x28+var_8($sp)
        sw      $s0, 0x28+var_C($sp)
        move   $fp, $sp
        sw      $a0, 0x28+arg_0($fp)
        sw      $a1, 0x28+arg_4($fp)
        lw      $v0, 0x28+arg_0($fp)
        or      $at, $zero
        bnez  $v0, loc_40
        or      $at, $zero
        lw      $v0, 0x28+arg_4($fp)
        or      $at, $zero
        addiu $v0, 1
        b      loc_AC
        or      $at, $zero

loc_40:                                # CODE XREF: f+24
        lw      $v0, 0x28+arg_4($fp)
        or      $at, $zero
        bnez  $v0, loc_74
        or      $at, $zero
        lw      $v0, 0x28+arg_0($fp)
        or      $at, $zero
        addiu $v0, -1
        move   $a0, $v0
        li      $a1, 1
        jal    f
        or      $at, $zero
        b      loc_AC
        or      $at, $zero

loc_74:                                # CODE XREF: f+48
        lw      $v0, 0x28+arg_0($fp)
        or      $at, $zero
        addiu $s0, $v0, -1
        lw      $v0, 0x28+arg_4($fp)
        or      $at, $zero
        addiu $v0, -1
        lw      $a0, 0x28+arg_0($fp)
        move   $a1, $v0
        jal    f
        or      $at, $zero
        move   $a0, $s0
        move   $a1, $v0
        jal    f
        or      $at, $zero

loc_AC:                                # CODE XREF: f+38

```

	# f+6C
move	\$sp, \$fp
lw	\$ra, 0x28+var_4(\$sp)
lw	\$fp, 0x28+var_8(\$sp)
lw	\$s0, 0x28+var_C(\$sp)
addiu	\$sp, 0x28
jr	\$ra
or	\$at, \$zero

100.8 Ejercicio 2.17

.?

:

- Linux x64 ([beginners.re](#))
- Mac OS X ([beginners.re](#))
- Linux MIPS ([beginners.re](#))
- Win32 ([beginners.re](#))
- Win64 ([beginners.re](#))

[MSVC 2012 redist.](#)

100.9 Ejercicio 2.18

..

..

.

- Win32 ([beginners.re](#))
- Linux x86 ([beginners.re](#))
- Mac OS X ([beginners.re](#))
- Linux MIPS ([beginners.re](#))

100.10 Ejercicio 2.19

2.18.

- Win32 ([beginners.re](#))

- Linux x86 ([beginners.re](#))
- Mac OS X ([beginners.re](#))
- Linux MIPS ([beginners.re](#))

100.11 Ejercicio 2.20

?

:

- Linux x64 ([beginners.re](#))
- Mac OS X ([beginners.re](#))
- Linux ARM Raspberry Pi ([beginners.re](#))
- Linux MIPS ([beginners.re](#))
- Win64 ([beginners.re](#))

Capítulo 101

3

101.1 Ejercicio 3.2

- Windows x86 ([beginners.re](#))
- Linux x86 ([beginners.re](#))
- Mac OS X (x64) ([beginners.re](#))
- Linux MIPS ([beginners.re](#))

101.2 Ejercicio 3.3

- Windows x86 ([beginners.re](#))
- Linux x86 ([beginners.re](#))
- Mac OS X (x64) ([beginners.re](#))
- Linux MIPS ([beginners.re](#))

101.3 Ejercicio 3.4

- Windows x86 (beginners.re)
- :<http://go.yurichev.com/17197>

101.4 Ejercicio 3.5

- . . .
:
• .
• .
• Windows x86 (beginners.re)
• Linux x86 (beginners.re)
• Mac OS X (x64) (beginners.re)
• Linux MIPS (beginners.re)
• :beginners.re

101.5 Ejercicio 3.6

Spanish text placeholder. . .

- Windows x86 (beginners.re)
- Linux x86 (beginners.re)
- Mac OS X (x64) (beginners.re)

101.6 Ejercicio 3.8

:beginners.re
:beginners.re
:beginners.re.
. . .

- Windows x86 (beginners.re)
- Linux x86 (beginners.re)

- Mac OS X (x64) (beginners.re)

Capítulo 102

crackme / keygenme

<http://go.yurichev.com/17315>

Capítulo 103

:<dennis(a)yurichev.com>

Spanish text placeholder.

[GitHub.](#)

Apéndice A

x86

A.1

byte 8-Spanish text placeholder. . :AL/BL/CL/DL/AH/BH/CH/DH/SIL/DIL/R*L.

word 16-Spanish text placeholder. . :AX/BX/CX/DX/SI/DI/R*W.

double word («dword») 32-Spanish text placeholder. . (x86) (x64). .

quad word («qword») 64-Spanish text placeholder. . .

tbyte (10) 80-Spanish text placeholder o 10 (IEEE 754 FPU).

paragraph (16) .

Windows API¹.

A.2

N.B.: .

¹Application programming interface

A.2.1 RAX/EAX/AX/AL

Spanish text placeholder	
RAX ^{x64}	
	EAX
	AX
AH	AL

AKA . .

A.2.2 RBX/EBX/BX/BL

Spanish text placeholder	
RBX ^{x64}	
	EBX
	BX
BH	BL

A.2.3 RCX/ECX/CX/CL

Spanish text placeholder	
RCX ^{x64}	
	ECX
	CX
CH	CL

AKA : (SHL/SHR/RxL/RxR).

A.2.4 RDX/EDX/DX/DL

Spanish text placeholder	
RDX ^{x64}	
	EDX
	DX
DH	DL

A.2.5 RSI/ESI/SI/SIL

Spanish text placeholder	
RSI ^{x64}	
	ESI
	SI
SIL	^{x64}

AKA «source index». REP MOVSx, REP CMPSx.

A.2.6 RDI/EDI/DI/DIL

Spanish text placeholder	
RD $I^{\times 64}$	
EDI	
DI	
DIL $^{\times 64}$	

AKA «destination index». REP MOVSx, REP STOSx.

A.2.7 R8/R8D/R8W/R8L

Spanish text placeholder	
R8	
R8D	
R8W	
R8L	

A.2.8 R9/R9D/R9W/R9L

Spanish text placeholder	
R9	
R9D	
R9W	
R9L	

A.2.9 R10/R10D/R10W/R10L

Spanish text placeholder	
R10	
R10D	
R10W	
R10L	

A.2.10 R11/R11D/R11W/R11L

Spanish text placeholder	
R11	
R11D	
R11W	
R11L	

A.2.11 R12/R12D/R12W/R12L

Spanish text placeholder	
R12	
R12D	
R12W	
R12L	

A.2.12 R13/R13D/R13W/R13L

Spanish text placeholder	
R13	
R13D	
R13W	
R13L	

A.2.13 R14/R14D/R14W/R14L

Spanish text placeholder	
R14	
R14D	
R14W	
R14L	

A.2.14 R15/R15D/R15W/R15L

Spanish text placeholder	
R15	
R15D	
R15W	
R15L	

A.2.15 RSP/ESP/SP/SPL

Spanish text placeholder	
RSP	
ESP	
SP	
SPL	

AKA stack pointer..

A.2.16 RBP/EBP/BP/BPL

Spanish text placeholder	
RBP	
EBP	
BP	
BPL	

AKA frame pointer. : ([7.1.2 on page 78](#)).

A.2.17 RIP/EIP/IP

Spanish text placeholder	
RIP ^{x64}	
EIP	
IP	

AKA «instruction pointer» ². :

```
MOV EAX, ...
JMP EAX
```

:

```
PUSH value
RET
```

A.2.18 CS/DS/ES/SS/FS/GS

(CS), (DS), (SS).

FS en win32 [TLS](#), . .
([94 on page 1160](#)).

A.2.19

AKA EFLAGS.

²«program counter»

0	0	
0 (1)	CF (Carry)	CLC/STC/CMC
2 (4)	PF (Parity)	(17.7.1 on page 288).
4 (0x10)	AF (Adjust)	
6 (0x40)	ZF (Zero)	0 0.
7 (0x80)	SF (Sign)	
8 (0x100)	TF (Trap)	.
9 (0x200)	IF (Interrupt enable)	.
9 (0x200)	IF (Interrupt enable)	CLI/STI
10 (0x400)	DF (Direction)	REP MOVSx, REP CMPSx, REP LI CLD/STD
11 (0x800)	OF (Overflow)	
12, 13 (0x3000)	IOPL (I/O privilege level) ⁸⁰²⁸⁶	
14 (0x4000)	NT (Nested task) ⁸⁰²⁸⁶	
16 (0x10000)	RF (Resume) ⁸⁰³⁸⁶	.
17 (0x20000)	VM (Virtual 8086 mode) ⁸⁰³⁸⁶	
18 (0x40000)	AC (Alignment check) ⁸⁰⁴⁸⁶	
19 (0x80000)	VIF (Virtual interrupt) ^{Pentium}	
20 (0x100000)	VIP (Virtual interrupt pending) ^{Pentium}	
21 (0x200000)	ID (Identification) ^{Pentium}	

A.3 FPU registros

8 80-: ST(0)-ST(7). N.B.: IDA ST(0) ST. IEEE 754.

:

79	78	64	63	62	0
S		I			

(S, I)

A.3.1

FPU.

0	IM (Invalid operation Mask)	
1	DM (Denormalized operand Mask)	
2	ZM (Zero divide Mask)	
3	OM (Overflow Mask)	
4	UM (Underflow Mask)	
5	PM (Precision Mask)	
7	IEM (Interrupt Enable Mask)	
8, 9	PC (Precision Control)	00 – 24 (REAL4) 10 – 53 (REAL8) 11 – 64 (REAL10)
10, 11	RC (Rounding Control)	00 – 01 – $-\infty$ 10 – $+\infty$ 11 – 0
12	IC (Infinity Control)	0 – 0 $+\infty$ y $-\infty$ 1 – $+\infty$ y $-\infty$

PM, UM, OM, ZM, DM, IM .

A.3.2

15	B (Busy)	(1) (0)
14	C3	
13, 12, 11	TOP	
10	C2	
9	C1	
8	C0	
7	IR (Interrupt Request)	
6	SF (Stack Fault)	
5	P (Precision)	
4	U (Underflow)	
3	O (Overflow)	
2	Z (Zero)	
1	D (Denormalized)	
0	I (Invalid operation)	

SF, P, U, O, Z, D, I .

C3, C2, C1, C0 : ([17.7.1 on page 288](#)).

N.B.:.

A.3.3 Tag Word

15, 14	Tag(7)
13, 12	Tag(6)
11, 10	Tag(5)
9, 8	Tag(4)
7, 6	Tag(3)
5, 4	Tag(2)
3, 2	Tag(1)
1, 0	Tag(0)

:

- 00 –
- 01 –
- 10 – ([NAN](#)³, ∞ , 0)
- 11 –

A.4 SIMD registros

A.4.1 MMX registros

8 64-: MM0..MM7.

A.4.2 SSE y AVX registros

SSE: 8 128-: XMM0..XMM7. x86-64 : XMM8..XMM15.

AVX .

A.5

hardware breakpoints.

³Not a Number

- DR0 – #1
- DR1 – #2
- DR2 – #3
- DR3 – #4
- DR6 –
- DR7 –

A.5.1 DR6

0	
0 (1)	B0 –
1 (2)	B1 –
2 (4)	B2 –
3 (8)	B3 –
13 (0x2000)	BD –
14 (0x4000)	BS – .
15 (0x8000)	BT (task switch flag)

N.B. . ([A.2.19 on page 1225](#)).

A.5.2 DR7

0	
0 (1)	L0 – #1
1 (2)	G0 – #1
2 (4)	L1 – #2
3 (8)	G1 – #2
4 (0x10)	L2 – #3
5 (0x20)	G2 – #3
6 (0x40)	L3 – #4
7 (0x80)	G3 – #4
8 (0x100)	LE –
9 (0x200)	GE –
13 (0x2000)	GD –
16,17 (0x30000)	#1: R/W –
18,19 (0xC0000)	#1: LEN –
20,21 (0x300000)	#2: R/W –
22,23 (0xC00000)	#2: LEN –
24,25 (0x3000000)	#3: R/W –
26,27 (0xC000000)	#3: LEN –
28,29 (0x30000000)	#4: R/W –
30,31 (0xC0000000)	#4: LEN –

(R/W):

- 00 –
- 01 –
- 10 –
- 11 –

N.B.: .

(LEN):

- 00 –
- 01 –
- 10 –
- 11 –

A.6

(90 on page 1145).

- . [Int13] o [AMD13a].
(89.2 on page 1144).

A.6.1

LOCK ... ADD, AND, BTR, BTS, CMPXCHG, OR, XADD, XOR. (68.4 on page 938).

REP MOVSx y STOSx: . MOVSx (A.6.2 on page 1234) y STOSx (A.6.2 on page 1236).

REPE/REPNE (AKA REPZ/REPNZ) CMPSx y SCASx: . .

CMPSx (A.6.3 on page 1237) y SCASx (A.6.2 on page 1235).

A.6.2

ADC (*add with carry*)

```

;
;
;
ADD val1.lo, val2.lo
ADC val1.hi, val2.hi ;

```

: 24 on page 519.

ADD

AND

CALL :PUSH address_after_CALL_instruction; JMP label

CMP

DEC *decrement*. .

IMUL

INC *increment*. .

JCXZ, JECXZ, JRCXZ (M)

JMP . *jump offset*.

Jcc (cccondition code)

. . *jump offset*.

JAE AKA JNC: : CF=0

JA AKA JNBE: : CF=0 y ZF=0

JBE : CF=1 o ZF=1

JB AKA JC: : CF=1

JC AKA JB:

JE AKA JZ: : ZF=1

JGE : SF=OF

JG : ZF=0 y SF=OF

JLE : ZF=1 o SF≠OF

JL : SF≠OF

JNAE AKA JC: CF=1

JNA CF=1 y ZF=1

JNBE : CF=0 y ZF=0

JNB AKA JNC: : CF=0

JNC AKA JAE: JNB.

JNE AKA JNZ: : ZF=0

JNGE : SF≠OF

JNG : ZF=1 o SF≠OF

JNLE : ZF=0 y SF=OF

JNL : SF=OF

JNO : OF=0

JNS

JNZ AKA JNE: : ZF=0

JO : OF=1

JPO (Jump Parity Odd)

JP AKA JPE:

JS

JZ AKA JE: : ZF=1

LAHF :

**LEAE** (*Load Effective Address*)

Spanish text placeholder

Spanish text placeholder

Spanish text placeholder.

```
int f(int a, int b)
{
    return a*8+b;
};
```

Listing A.1: Con optimización MSVC 2010

```
_a$ = 8 ; ↴
↳ size = 4
_b$ = 12 ; ↴
↳ size = 4
_f      PROC
        mov     eax, DWORD PTR _b$[esp-4]
        mov     ecx, DWORD PTR _a$[esp-4]
        lea     eax, DWORD PTR [eax+ecx*8]
        ret     0
_f      ENDP
```

Intel C++ :

```
int f1(int a)
{
    return a*13;
};
```

Listing A.2: Intel C++ 2011

```
_f1      PROC NEAR
        mov     ecx, DWORD PTR [4+esp]      ; ecx = a
        lea     edx, DWORD PTR [ecx+ecx*8]   ; edx = a*9
        lea     eax, DWORD PTR [edx+ecx*4]   ; eax = a*9 ↴
        ↳ + a*4 = a*13
        ret
```

MOVSB/MOVSW/MOVSD/MOVSQ / 16-/ 32-/ 64- SI/ESI/RSI DI/EDI/RDI.

`memcpy() . .`

`memcpy(EDI, ESI, 15):`

```
;  
CLD ;  
MOV ECX, 3 ;  
REP MOVSD ;  
MOVSW ;  
MOVSB ;
```

`(.).`

MOVSX : ([15.1.1 on page 246](#))

MOVZX : ([15.1.1 on page 247](#))

MOV .

MUL

NEG : $op = -op$

NOP **NOP**. XCHG EAX, EAX. . [6.1.1 on page 53](#).

: ([88 on page 1141](#)).

NOP . **NOP** .

NOT op1: $op1 = \neg op1$.

OR

POP : value=SS:[ESP]; ESP=ESP+4 (o 8)

PUSH : ESP=ESP-4 (o 8); SS:[ESP]=value

RET : POP tmp; JMP tmp.

, RETN («return near») ([94 on page 1160](#)), RETF («return far»).

: POP tmp; ADD ESP op1; JMP tmp. RET : [64.2 on page 869](#).

SAHF :



SBB (*subtraction with borrow*)

```

;
;
SUB val1.lo, val2.lo
SBB val1.hi, val2.hi ;

```

: 24 on page 519.

SCASB/SCASW/SCASD/SCASQ (M) / 16-/ 32-/ 64- AX/EAX/RAX DI/EDI/RDI..

```

...
:
:
lea      edi, string
mov      ecx, 0FFFFFFFh ;
xor      eax, eax        ;
repne scasb
add      edi, 0FFFFFFFh ;

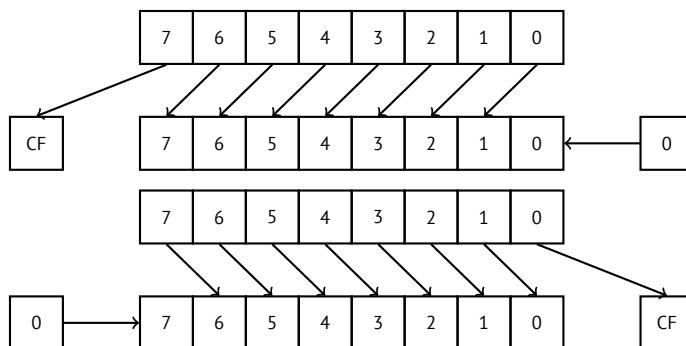
;
;

;   ECX = -1-strlen

not      ecx
dec      ecx
;
```

SHL

SHR :



2^n . : 19 on page 389.

SHRD op1, op2, op3: .

: 24 on page 519.

STOSB/STOSW/STOSD/STOSQ / 16-/ 32-/ 64- AX/EAX/RAX DI/EDI/RDI.

: memset() . .

memset(EDI, 0xAA, 15):

```
;  
CLD ;  
MOV EAX, 0AAAAAAAhh  
MOV ECX, 3  
REP STOSD ;  
STOSW ;  
STOSB ;
```

(.).

SUB

TEST : 19 on page 389

XCHG

XOR op1, op2: **XOR**⁴. $op1 = op1 \oplus op2$.

0	0	0
0	1	1
1	0	1
1	1	0

A.6.3

BSF bit scan forward, : 25.2 on page 548

BSR bit scan reverse

BSWAP (byte swap), .

BTC bit test and complement

BTR bit test and reset

BTS bit test and set

⁴eXclusive OR

BT bit test**CBW/CWD/CWDE/CDQ/CDQE :****CBW****CWD****CWDE****CDQ****CDQE (x64)**[24.5 on page 531.](#)

The process of stretching numbers by extending the sign bit is called sign extension. The 8086 provides instructions (Fig. 3.29) to facilitate the task of sign extension. These instructions were initially named SEX (sign extend) but were later renamed to the more conservative CBW (convert byte to word) and CWD (convert word to double word).

CLD .**CLI (M)****CMC (M)****CMOVcc** MOV:.. ([A.6.2 on page 1231](#)).**CMPSB/CMPSW CMPSD/CMPSQ (M) / 16-/ 32-/ 64- SI/ESI/RSI DI/EDI/RDI..**

memcmp() .

(WRK v1.2):

Listing A.3: base\ntos\rtl\i386\movemem.asm

```
; ULONG
; RtlCompareMemory (
;     IN PVOID Source1,
;     IN PVOID Source2,
;     IN ULONG Length
; )
;
; Routine Description:
;
;     This function compares two blocks of memory and ↴
;         returns the number
```

```

;      of bytes that compared equal.

; Arguments:
;
;      Source1 (esp+4) - Supplies a pointer to the first ↴
;      ↴ block of memory to
;      ↴ compare.
;
;      Source2 (esp+8) - Supplies a pointer to the second ↴
;      ↴ block of memory to
;      ↴ compare.
;
;      Length (esp+12) - Supplies the Length, in bytes, of ↴
;      ↴ the memory to be
;      ↴ compared.
;
; Return Value:
;
;      The number of bytes that compared equal is returned ↴
;      ↴ as the function
;      value. If all bytes compared equal, then the length ↴
;      ↴ of the original
;      block of memory is returned.
;
;---

RcmSource1      equ      [esp+12]
RcmSource2      equ      [esp+16]
RcmLength       equ      [esp+20]

CODE_ALIGNMENT
cPublicProc _RtlCompareMemory,3
cPublicFpo 3,0

    push    esi          ; save registers
    push    edi          ;
    cld               ; clear direction
    mov     esi,RcmSource1 ; (esi) -> first ↴
    ↴ block to compare
    mov     edi,RcmSource2 ; (edi) -> second ↴
    ↴ block to compare

;
; Compare dwords, if any.
;

rcm10:  mov     ecx,RcmLength           ; (ecx) = length ↴

```

```

    ↳ in bytes
        shr      ecx,2           ; (ecx) = length ↵
    ↳ in dwords
        jz       rcm20          ; no dwords, try ↵
    ↳ bytes
        repe    cmpsd           ; compare dwords
        jnz     rcm40          ; mismatch, go ↵
    ↳ find byte

;

; Compare residual bytes, if any.
;

rcm20: mov      ecx,RcmLength          ; (ecx) = length ↵
    ↳ in bytes
        and     ecx,3           ; (ecx) = length ↵
    ↳ mod 4
        jz      rcm30          ; 0 odd bytes, go ↵
    ↳ do dwords
        repe   cmpsb           ; compare odd ↵
    ↳ bytes
        jnz    rcm50          ; mismatch, go ↵
    ↳ report how far we got

;

; All bytes in the block match.
;

rcm30: mov      eax,RcmLength          ; set number of ↵
    ↳ matching bytes
        pop     edi             ; restore ↵
    ↳ registers
        pop     esi             ;
        stdRET _RtlCompareMemory

;

; When we come to rcm40, esi (and edi) points to the ↵
; dword after the
; one which caused the mismatch. Back up 1 dword and ↵
; find the byte.
; Since we know the dword didn't match, we can assume ↵
; one byte won't.
;

rcm40: sub     esi,4           ; back up
        sub     edi,4           ; back up
        mov     ecx,5           ; ensure that ecx ↵

```

```

    ↳ doesn't count out
      repe    cmpsb           ; find mismatch ↵
    ↳ byte

;

; When we come to rcm50, esi points to the byte after ↵
; the one that
; did not match, which is TWO after the last byte that ↵
; did match.
;

rcm50: dec     esi          ; back up
       sub     esi,RcmSource1   ; compute bytes ↵
    ↳ that matched
      mov     eax,esi          ;
      pop     edi              ; restore ↵
    ↳ registers
      pop     esi              ;
      stdRET _RtlCompareMemory

stdENDP _RtlCompareMemory

```

N.B.: .

CPUID . : ([21.6.1 on page 477](#)).

DIV

IDIV

INT (M): INT x PUSHF; CALL dword ptr [x*4]

. : [[Bro](#)] .

INT 3 (M): INT, (0xCC), . 0xCC .

[Windows NT](#), [EXCEPTION_BREAKPOINT](#) . . . [MSVS](#)⁵ . reverse engineering,

[MSVC compiler intrinsic](#) : `__debugbreak()`⁶.

`kernel32.dll DebugBreak()`⁷, INT 3.

IN (M) . ([78.3 on page 1001](#)).

IRET . . POP tmp; POPF; JMP tmp.

⁵Microsoft Visual Studio

⁶[MSDN](#)

⁷[MSDN](#)

LOOP (M) CX/ECX/RCX, .

OUT (M) . ([78.3 on page 1001](#)).

POPA (M) (R|E)DI, (R|E)SI, (R|E)BP, (R|E)BX, (R|E)DX, (R|E)CX, (R|E)AX .

POPCNT population count. . AKA «hamming weight». AKA «NSA instruction» :

This branch of cryptography is fast-paced and very politically charged. Most designs are secret; a majority of military encryptions systems in use today are based on LFSRs. In fact, most Cray computers (Cray 1, Cray X-MP, Cray Y-MP) have a rather curious instruction generally known as “population count.” It counts the 1 bits in a register and can be used both to efficiently calculate the Hamming distance between two binary words and to implement a vectorized version of a LFSR. I’ve heard this called the canonical NSA instruction, demanded by almost all computer contracts.

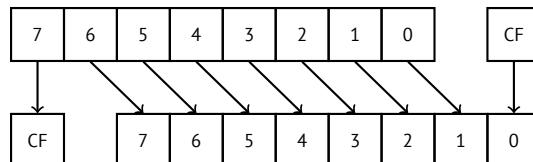
[Sch94]

POPF (AKA)

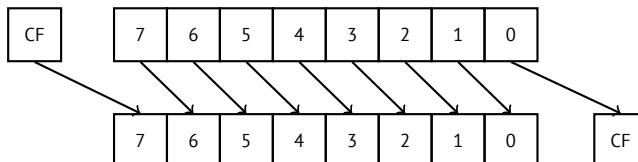
PUSHA (M) (R|E)AX, (R|E)CX, (R|E)DX, (R|E)BX, (R|E)BP, (R|E)SI, (R|E)DI .

PUSHF (AKA)

RCL (M) :

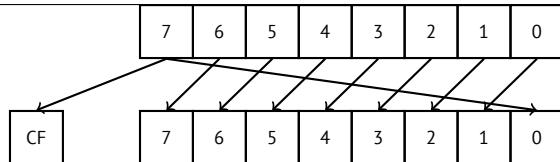


RCR (M) :

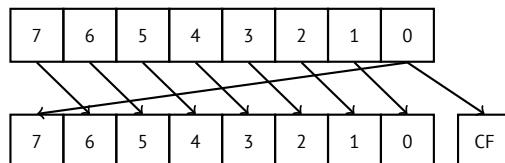
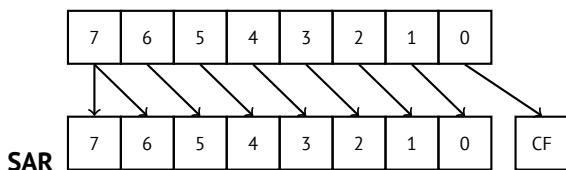


ROL/ROR (M)

ROL::



ROR::

(compiler intrinsics) `_rotl()` y `_rotr()`⁸, .**SAL , SHL**

MSB.

SETcc op: . (A.6.2 on page 1231).**STC** (M)**STD** (M). ntoskrnl.exe.**STI** (M)**SYSCALL** (AMD) (66 on page 886)**SYSENTER** (Intel) (66 on page 886)**UD2** (M)

A.6.4

-P

FABS⁸ MSDN

FADD op: ST(0)=op+ST(0)**FADD** ST(0), ST(i): ST(0)=ST(0)+ST(i)**FADDP** ST(1)=ST(0)+ST(1);**FCHS** ST(0)=-ST(0)**FCOM** ST(0) ST(1)**FCOM** op: ST(0) op**FCOMP** ST(0) ST(1);**FCOMPP** ST(0) ST(1);**FDIVR** op: ST(0)=op/ST(0)**FDIVR** ST(i), ST(j): ST(i)=ST(j)/ST(i)**FDIVRP** op: ST(0)=op/ST(0);**FDIVRP** ST(i), ST(j): ST(i)=ST(j)/ST(i);**FDIV** op: ST(0)=ST(0)/op**FDIV** ST(i), ST(j): ST(i)=ST(i)/ST(j)**FDIVP** ST(1)=ST(0)/ST(1);**FILD** op: .**FIST** op: ST(0) op**FISTP** op: ST(0) op;**FLD1****FLDCW** op: FPU control word ([A.3 on page 1226](#)) 16-bit op.**FLDZ****FLD** op: .**FMUL** op: ST(0)=ST(0)*op**FMUL** ST(i), ST(j): ST(i)=ST(i)*ST(j)**FMULP** op: ST(0)=ST(0)*op;**FMULP** ST(i), ST(j): ST(i)=ST(i)*ST(j);**FSINCOS** : tmp=ST(0); ST(1)=sin(tmp); ST(0)=cos(tmp)**FSQRT** : $ST(0) = \sqrt{ST(0)}$ **FSTCW** op: FPU control word ([A.3 on page 1226](#)) 16-bit op .

FNSTCW op: FPU control word ([A.3 on page 1226](#)) 16-bit op.

FSTSW op: FPU status word ([A.3.2 on page 1227](#)) 16-bit op.

FNSTSW op: FPU status word ([A.3.2 on page 1227](#)) 16-bit op.

FST op: ST(0) op

FSTP op: ST(0) op;

FSUBR op: ST(0)=op-ST(0)

FSUBR ST(0), ST(i): ST(0)=ST(i)-ST(0)

FSUBRP ST(1)=ST(0)-ST(1);

FSUB op: ST(0)=ST(0)-op

FSUB ST(0), ST(i): ST(0)=ST(0)-ST(i)

FSUBP ST(1)=ST(1)-ST(0);

FUCOM ST(i): ST(0) y ST(i)

FUCOM ST(0) y ST(1)

FUCOMP ST(0) y ST(1);.

FUCOMPP ST(0) y ST(1);.

FXCH ST(i)

FXCH

A.6.5

()

: [82.1 on page 1089](#).

ASCII	x86
0	XOR
1	XOR
2	XOR
3	XOR
4	XOR
5	XOR
7	AAA
8	CMP
9	CMP

:	3a	CMP
;	3b	CMP
<	3c	CMP
=	3d	CMP
?	3f	AAS
@	40	INC
A	41	INC
B	42	INC
C	43	INC
D	44	INC
E	45	INC
F	46	INC
G	47	INC
H	48	DEC
I	49	DEC
J	4a	DEC
K	4b	DEC
L	4c	DEC
M	4d	DEC
N	4e	DEC
O	4f	DEC
P	50	PUSH
Q	51	PUSH
R	52	PUSH
S	53	PUSH
T	54	PUSH
U	55	PUSH
V	56	PUSH
W	57	PUSH
X	58	POP
Y	59	POP
Z	5a	POP
[5b	POP
\	5c	POP
]	5d	POP
^	5e	POP
-	5f	POP
a	60	PUSHA
f	61	POPA
	66	
g	67	

h	68	PUSH
i	69	IMUL
j	6a	PUSH
k	6b	IMUL
p	70	JO
q	71	JNO
r	72	JB
s	73	JAE
t	74	JE
u	75	JNE
v	76	JBE
w	77	JA
x	78	JS
y	79	JNS
z	7a	JP

: AAA, AAS, CMP, DEC, IMUL, INC, JA, JAE, JB, JBE, JE, JNE, JNO, JNS, JO, JP, JS, POP, POPA, PUSH, PUSH, XOR.

Apéndice B

ARM

B.1

ARM [CPU](#), .

byte 8-Spanish text placeholder. .

halfword 16-Spanish text placeholder. .

word 32-Spanish text placeholder. .

doubleword 64-Spanish text placeholder.

quadword 128-Spanish text placeholder.

B.2

- ARMv4: .
- ARMv6: iPhone 1st gen., iPhone 3G (Samsung 32-bit RISC ARM 1176JZ(F)-S Thumb-2)
- ARMv7: Thumb-2 (2003). iPhone 3GS, iPhone 4, iPad 1st gen. (ARM Cortex-A8), iPad 2 (Cortex-A9), iPad 3rd gen.
- ARMv7s: . iPhone 5, iPhone 5c, iPad 4th gen. (Apple A6).
- ARMv8: 64-, [AKA](#) ARM64 [AKA](#) AArch64. iPhone 5S, iPad Air (Apple A7).

B.3 32- ARM (AArch32)

B.3.1

- R0
- R1...R12 GPRs
- R13 AKA SP (*stack pointer*)
- R14 AKA LR (*link register*)
- R15 AKA PC (*program counter*)

R0-R3 .

B.3.2 Current Program Status Register (CPSR)

0..4	Mprocessor mode
5	TThumb state
6	FFIQ disable
7	IIRQ disable
8	Aimprecise data abort disable
9	Edata endianness
10..15, 25, 26	ITif-then state
16..19	GEgreater-than-or-equal-to
20..23	DNMdo not modify
24	JJava state
27	Qsticky overflow
28	Voverflow
29	Ccarry/borrow/extend
30	Zzero bit
31	Nnegative/less than

B.3.3

0..31 ^{bits}	32..64	65..96	97..127
Q0 ^{128 bits}			
D0 ^{64 bits}		D1	
S0 ^{32 bits}	S1	S2	S3

• VFPv3 (NEON o «Advanced SIMD») .

NEON o «Advanced SIMD» .

B.4 64- ARM (AArch64)

B.4.1

AArch32.

- X0
- X0...X7.
- X8
- X9...X15.
- X16
- X17
- X18
- X19...X29.
- X29 FP (GCC)
- X30«Procedure Link Register» AKA LR (link register).
- X31 AKA XZR o «Zero Register». WZR.
- SP, .

: [ARM13c].

Spanish text placeholder	Spanish text placeholder
	X0
	W0

B.5

ADD ADDS .

B.5.1

<u>EQ</u>		Z == 1
NE		Z == 0
CS AKA HS (Higher or Same)	/	C == 1
CC AKA LO (LOwer)	/	C == 0
MI	/	N == 1
PL	/	N == 0
VS		V == 1
VC		V == 0
HI	/	C == 1 and Z == 0
LS	/	C == 0 or Z == 1
GE	/	N == V
LT	/	N != V
GT	/	Z == 0 and N == V
LE	/	Z == 1 or N != V
None / AL		

Apéndice C

MIPS

C.1 Registros

()

C.1.1 GPR

\$0	\$ZERO	(NOP).
\$1	\$AT	.
\$2 ...\$3	\$V0 ...\$V1	.
\$4 ...\$7	\$A0 ...\$A3	.
\$8 ...\$15	\$T0 ...\$T7	.
\$16 ...\$23	\$S0 ...\$S7	*.
\$24 ...\$25	\$T8 ...\$T9	.
\$26 ...\$27	\$K0 ...\$K1	.
\$28	\$GP	**.
\$29	\$SP	SP*.
\$30	\$FP	FP*.
\$31	\$RA	RA.
n/a	PC	PC.
n/a	HI	***.
n/a	LO	***.

C.1.2

\$F0..\$F1	.
\$F2..\$F3	.
\$F4..\$F11	.
\$F12..\$F15	.
\$F16..\$F19	.
\$F20..\$F31	*.

*Callee .

**Callee () .

*** MFHI y MFLO.

C.2 Instrucciones

:

- instruction destination, source1, source2

instruction destination/source1, source2
--

-
-

C.2.1

Apéndice D

__divdi3	
__moddi3	
__udivdi3	
__umoddi3	

Apéndice E

11 «long long», .

__alldiv	
__allmul	
__allrem	
__allshl	
__allshr	
__aulldiv	
__aullrem	
__aullshr	

VC/crt/src/intel/*.asm.

Apéndice F

Cheatsheets

F.1 IDA

Cheatsheet de teclas de acceso rápido:

Space	
C	
D	
A	
*	
U	
O	
H	
R	
B	
Q	
N	
?	
G	
:	
Ctrl-X	
X	Spanish text placeholder.
Alt-I	
Ctrl-I	
Alt-B	
Ctrl-B	
Alt-T	
Ctrl-T	
Alt-P	
Enter	
Esc	
Num -	
Num +	

F.2 OllyDbg

Cheatsheet de teclas de acceso rápido:

F7		
F8		
F9		
Ctrl-F2		pasar por encima

F.3 MSVC

/O1	
/Ob0	
/Ox	
/GS-	
/Fa(file)	
/Zi	
/Zp(n)	
/MD	MSVCR*.DLL

: 55.1 on page 842.

F.4 GCC

-Os	
-O3	
-regparm=	
-o file	
-g	
-S	
-masm=intel	
-fno-inline	

F.5 GDB

:

break filename.c:number	
break function	
break *address	
b	—”—
p variable	
run	
r	—”—
cont	
c	—”—
bt	
set disassembly-flavor intel	
disas	disassemble current function
disas function	
disas function,+50	disassemble portion
disas \$eip,+0x10	—”—
disas/r	
info registers	
info float	
info locals	
x/w ...	
x/w \$rdi	
x/10w ...	
x/s ...	
x/i ...	
x/10c ...	
x/b ...	
x/h ...	
x/g ...	
finish	
next	
step	
set step-mode on	
frame n	
info break	
del n	
set args ...	

Apéndice G

G.1

G.1.1

Ejercicio #1

Ejercicio: [3.7.1 on page 30.](#)

```
MessageBeep (0xFFFFFFFF); // A simple beep. If the sound card ↵
    ↵ is not available, the sound is generated using the ↵
    ↵ speaker.
```

Ejercicio #2

Ejercicio: [3.7.2 on page 31.](#)

```
#include <unistd.h>

int main()
{
    sleep(2);
}
```

G.1.2

Ejercicio #1

Ejercicio: [5.5.1 on page 43.](#)

, RA y argc..

: RA, argc argv[].

GCC 4.8.x .

Ejercicio #2

Ejercicio: [5.5.2 on page 43.](#)

```
#include <stdio.h>
#include <time.h>

int main()
{
    printf ("%d\n", time(NULL));
}
```

G.1.3

Ejercicio #1

Ejercicio: [7.4.1 on page 112.](#)

("read only data"):

```
$ objdump -s 1

...
Contents of section .rodata:
400600 01000200 52657375 6c743a20 25730a00 ...Result: %s..
400610 48656c6c 6f2c2077 6f726c64 210a00 Hello, world!..
```

C:\...>objdump -s 1.exe

```
...
Contents of section .data:
40b000 476f6f64 62796521 00000000 52657375 Goodbye!....Resu
40b010 6c743a20 25730a00 48656c6c 6f2c2077 lt: %s..Hello, w
40b020 6f726c64 210a0000 00000000 00000000 orld!.....@.....
40b030 01000000 00000000 c0cb4000 00000000 .....@.....
```

...

C:\...>objdump -x 1.exe

...

Sections:						
Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	00006d2a	00401000	00401000	00000400	2**2
		CONTENTS, ALLOC, LOAD, READONLY, CODE				
1	.rdata	00002262	00408000	00408000	00007200	2**2
		CONTENTS, ALLOC, LOAD, READONLY, DATA				
2	.data	00000e00	0040b000	0040b000	00009600	2**2
		CONTENTS, ALLOC, LOAD, DATA				
3	.reloc	00000b98	0040e000	0040e000	0000a400	2**2
		CONTENTS, ALLOC, LOAD, READONLY, DATA				

G.1.4

G.1.5 Ejercicio #1

Ejercicio: [13.5.1 on page 218.](#)

:printf().

G.1.6

G.1.7 Ejercicio #3

Ejercicio: [14.4.3 on page 239.](#)

```
#include <stdio.h>

int main()
{
    int i;
    for (i=100; i>0; i--)
        printf ("%d\n", i);
}
```

G.1.8 Ejercicio #4

Ejercicio: [14.4.4 on page 241.](#)

```
#include <stdio.h>

int main()
{
    int i;
    for (i=1; i<100; i=i+3)
        printf ("%d\n", i);
```

{};

G.1.9

Ejercicio #1

Ejercicio: [15.2.1 on page 256.](#)

```
int f(char *s)
{
    int rt=0;
    for (;*s;s++)
    {
        if (*s==' ')
            rt++;
    };
    return rt;
};
```

G.1.10

Ejercicio #2

Ejercicio: [16.3.1 on page 268.](#)

```
int f(int a)
{
    return a*7;
};
```

G.1.11

Ejercicio #1

Ejercicio: [17.10.2 on page 323.](#)

```
double f(double a1, double a2, double a3, double a4, double a5)
{
    return (a1+a2+a3+a4+a5) / 5;
};
```

G.1.12**Ejercicio #1**

Ejercicio: [18.9.1 on page 369.](#)

:

```
#define M      100
#define N      200

void s(double *a, double *b, double *c)
{
    for(int i=0;i<N;i++)
        for(int j=0;j<M;j++)
            *(c+i*M+j)=*(a+i*M+j) + *(b+i*M+j);
}
```

Ejercicio #2

Ejercicio: [18.9.2 on page 374.](#)

:

```
#define M      100
#define N      200
#define P      300

void m(double *a, double *b, double *c)
{
    for(int i=0;i<M;i++)
        for(int j=0;j<P;j++)
    {
        *(c+i*M+j)=0;
        for (int k=0;k<N;k++) *(c+i*M+j)+=*(a+i*M+j) * *(b+i*M+j);
    }
}
```

Ejercicio #3

Ejercicio: [18.9.3 on page 380.](#)

```
double f(double array[50][120], int x, int y)
{
    return array[x][y];
}
```

Ejercicio #4

Ejercicio: [18.9.4 on page 381.](#)

```
int f(int array[50][60][80], int x, int y, int z)
{
    return array[x][y][z];
}
```

Ejercicio #5

Ejercicio: [18.9.5 on page 383.](#)

```
int tbl[10][10];

int main()
{
    int x, y;
    for (x=0; x<10; x++)
        for (y=0; y<10; y++)
            tbl[x][y]=x*y;
}
```

G.1.13**Ejercicio #1**

Ejercicio: [19.7.1 on page 430.](#) endianness .

```
unsigned int f(unsigned int a)
{
    return ((a>>24)&0xff) | ((a<<8)&0xff0000) | ((a>>8)&0xff00) |
    | ((a<<24)&0xff000000);
}
```

Ejercicio #2

Ejercicio: [19.7.2 on page 432.](#)

```
#include <stdio.h>

unsigned int f(unsigned int a)
{
    int i=0;
    int j=1;
    unsigned int rt=0;
```

```

        for (;i<=28; i+=4, j*=10)
            rt+=((a>>i)&0xF) * j;
        return rt;
};

int main()
{
    // test
    printf ("%d\n", f(0x12345678));
    printf ("%d\n", f(0x1234567));
    printf ("%d\n", f(0x123456));
    printf ("%d\n", f(0x12345));
    printf ("%d\n", f(0x1234));
    printf ("%d\n", f(0x123));
    printf ("%d\n", f(0x12));
    printf ("%d\n", f(0x1));
};

```

Ejercicio #3

Ejercicio: [19.7.3 on page 435](#).

```
#include <windows.h>

int main()
{
    MessageBox(NULL, "hello, world!", "caption",
               MB_TOPMOST | MB_ICONINFORMATION | MB_HELP | ↴
               MB_YESNOCANCEL);
};
```

Ejercicio #4

Ejercicio: [19.7.4 on page 436](#).

```
#include <stdio.h>
#include <stdint.h>

// source code taken from
// http://www4.wittenberg.edu/academics/mathcomp/shelburne/ ↴
// comp255/notes/binarymultiplication.pdf

uint64_t mult (uint32_t m, uint32_t n)
{
    uint64_t p = 0; // initialize product p to 0
    while (n != 0) // while multiplier n is not 0
```

```

    {
        if (n & 1) // test LSB of multiplier
            p = p + m; // if 1 then add multiplicand m
        m = m << 1; // left shift multiplicand
        n = n >> 1; // right shift multiplier
    }
    return p;
}

int main()
{
    printf ("%d\n", mult (2, 7));
    printf ("%d\n", mult (3, 11));
    printf ("%d\n", mult (4, 111));
}

```

6.1.14

Ejercicio #1

Ejercicio: [21.7.1 on page 486](#).

```

#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    struct stat sb;

    if (argc != 2)
    {
        fprintf(stderr, "Usage: %s <pathname>\n", argv[0]);
        return 0;
    }

    if (stat(argv[1], &sb) == -1)
    {
        // error
        return 0;
    }

    printf("%ld\n", (long) sb.st_uid);
}

```

{}

Ejercicio #1

Ejercicio: [21.7.2 on page 487.](#)

```
#include <stdio.h>

struct some_struct
{
    int a;
    unsigned int b;
    float f;
    double d;
    char c;
    unsigned char uc;
};

void f(struct some_struct *s)
{
    if (s->a > 1000)
    {
        if (s->b > 10)
        {
            printf ("%f\n", s->f * 444 + s->d * ↴
                   123);
            printf ("%c, %d\n", s->c, s->uc);
        }
        else
        {
            printf ("error #2\n");
        }
    }
    else
    {
        printf ("error #1\n");
    }
}
```

G.1.15

Ejercicio #1

Ejercicio: [50.5.1 on page 699.](#)

:beginners.re.

G.1.16

Ejercicio #1

Ejercicio: [41.6 on page 638.](#)

```
int f(int a)
{
    return a/661;
}
```

G.2 1

G.2.1 Ejercicio 1.1

:

G.2.2 Ejercicio 1.4

[:beginners.re](#)

G.3 2

G.3.1 Ejercicio 2.1

Spanish text placeholder: [beginners.re](#)

G.3.2 Ejercicio 2.4

[:strstr\(\).](#)

:

```
char * strstr (
    const char * str1,
    const char * str2
)
{
    char *cp = (char *) str1;
    char *s1, *s2;

    if ( !*str2 )
        return((char *)str1);

    while (*cp)
```

```

{
    s1 = cp;
    s2 = (char *) str2;

    while ( *s1 && *s2 && !(*s1-*s2) )
        s1++, s2++;

    if (!*s2)
        return(cp);

    cp++;
}
return(NULL);
}

```

6.3.3 Ejercicio 2.6

([wikipedia](#)):

```

void f (unsigned int* v, unsigned int* k) {
    unsigned int v0=v[0], v1=v[1], sum=0, i;                      /* set */
    ↴ up */
    unsigned int delta=0x9e3779b9;                                /* a key */
    ↴ schedule constant */
    unsigned int k0=k[0], k1=k[1], k2=k[2], k3=k[3];      /* cache */
    ↴ key */
    for (i=0; i < 32; i++) {                                     /* basic */
    ↴ cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }
    ↴ */
    v[0]=v0; v[1]=v1;
}

```

6.3.4 Ejercicio 2.13

linear feedback shift register ¹.

Spanish text placeholder: [beginners.re](#)

¹[wikipedia](#)

G.3.5 Ejercicio 2.14

Spanish text placeholder: [beginners.re](#)

G.3.6 Ejercicio 2.15

Spanish text placeholder: [beginners.re](#)

G.3.7 Ejercicio 2.16

²

```
int ack (int m, int n)
{
    if (m==0)
        return n+1;
    if (n==0)
        return ack (m-1, 1);
    return ack(m-1, ack (m, n-1));
};
```

G.3.8 Ejercicio 2.17

:
[wikipedia](#).

Spanish text placeholder: [beginners.re](#)

G.3.9 Ejercicio 2.18

Spanish text placeholder: [beginners.re](#)

G.3.10 Ejercicio 2.19

Spanish text placeholder: [beginners.re](#)

²[wikipedia](#)

G.3.11 Ejercicio 2.20

: the On-Line Encyclopedia of Integer Sequences (OEIS)).

: *hailstone numbers*,³.

Spanish text placeholder: [beginners.re](#)

G.4 3

G.4.1 Ejercicio 3.2

:

:

[beginners.re](#)

G.4.2 Ejercicio 3.3

:

[beginners.re](#)

G.4.3 Ejercicio 3.4

[beginners.re](#)

G.4.4 Ejercicio 3.5

:

:

[beginners.re](#)

G.4.5 Ejercicio 3.6

:

[beginners.re](#)

G.4.6 Ejercicio 3.8

:

[beginners.re](#)

G.5

G.5.1 «Buscaminas (Windows XP)»

: 76 on page 958.

:

Lista de acrónimos usados

OS Sistema Operativo	XXX
PL Lenguaje de programación	4
ROM Memoria de solo lectura	857
RA Dirección de retorno	52
PE Portable Executable: 68.2 on page 899	
SP stack pointer. SP/ESP/RSP en x86/x64. SP en ARM.	18
DLL Dynamic-link library	899
PC Program Counter. IP/EIP/RIP en x86/64. PC en ARM.	18
LR Link Register	8
IDA Desensamblador y depurador interactivo desarrollado por Hex-Rays	8
IAT Import Address Table	900
INT Import Name Table	900
RVA Relative Virtual Address	900
VA Virtual Address	900
OEP Original Entry Point	899
MSVC Microsoft Visual C++	343
MSVS Microsoft Visual Studio	1240

ASLR Address Space Layout Randomization	901
MFC Microsoft Foundation Classes	904
TLS Thread Local Storage	xxviii
AKA Also Known As (También conocido como)	588
CRT C runtime library : 68.1 on page 895	19
CPU Central processing unit	xxx
FPU Floating-point unit	321
RISC Reduced instruction set computing	5
GUI Graphical user interface	901
RTTI Run-time type information	865
BSS Block Started by Symbol	902
SIMD Single instruction, multiple data	533
BSOD Black Screen of Death	886
ISA Instruction Set Architecture	iv
ELF Formato de archivo ejecutable usado ampliamente en sistemas *NIX, incluido Linux	xxviii
TIB Thread Information Block	345
PIC Position Independent Code: 67.1 on page 888	xxviii

NAN Not a Number	1228
NOP No OPeration	108
BEQ (PowerPC, ARM) Branch if Equal	110
BNE (PowerPC, ARM) Branch if Not Equal	977
BLR (PowerPC) Branch to Link Register	977
XOR eXclusive OR	1236
API Application programming interface	1221
ASCII American Standard Code for Information Interchange	1094
ASCIIZ ASCII Zero ()	107
IA64 Intel Architecture 64 (Itanium): 93 on page 1156	587
MSB Most significant bit/byte	403
HDD Hard disk drive	763
WRK Windows Research Kernel	863
GPR General Purpose Registers	xxvi
BOM Byte order mark	852
GDB GNU debugger	53
FP Frame Pointer	24

MBR Master Boot Record	857
JPE Jump Parity Even ()	293
STMFD Store Multiple Full Descending ()	
LDMFD Load Multiple Full Descending ()	
STMED Store Multiple Empty Descending ()	33
LDMED Load Multiple Empty Descending ()	33
STMFA Store Multiple Full Ascending ()	33
LDMFA Load Multiple Full Ascending ()	33
STMEA Store Multiple Empty Ascending ()	33
LDMEA Load Multiple Empty Ascending ()	33
APSR (ARM) Application Program Status Register	321
FPSCR (ARM) Floating-Point Status and Control Register	318
PID ID de programa/proceso	964
LF Line feed (10 o\n' en C/C++)	676
CR Carriage return (13 o\r' en C/C++)	676
TOS Top Of Stack	794
LVA (Java) Local Variable Array	801

JVM Java virtual machine	801
------------------------------------	-----

Glosario

Spanish text placeholder Spanish text placeholder. [18](#), [219](#), [575](#), [1231](#)

Spanish text placeholder Spanish text placeholder. [19](#), [575](#), [1231](#)

Spanish text placeholder Spanish text placeholder. [565](#)

Spanish text placeholder Spanish text placeholder. [777](#), [1224](#), [1248](#)

Spanish text placeholder Spanish text placeholder. [267](#), [274](#), [275](#), [564](#)

atomic operation «*ατομος*» Spanish text placeholder. [1152](#)

callee Spanish text placeholder. [1252](#)

caller Spanish text placeholder. [7–9](#)

compiler intrinsic Spanish text placeholder ([90](#) on page [1145](#)). [1240](#)

CP/M Control Program for Microcomputers: Spanish text placeholder MS-DOS.
[1090](#)

endianness Spanish text placeholder: [31](#) on page [586](#). [91](#), [1264](#)

heap Spanish text placeholder. [900](#)

jump offset Spanish text placeholder. [1231](#)

leaf function Spanish text placeholder. [35](#)

link register (RISC) Spanish text placeholder. [35](#), [977](#), [1248](#), [1249](#)

name mangling Spanish text placeholder: [51.1.1](#) on page [702](#). [842](#), [843](#)

NEON AKA «Advanced SIMD»SIMD Spanish text placeholder ARM. [1249](#)

PDB (Win32) Spanish text placeholder. [903, 958](#)

reverse engineering Spanish text placeholder. [1240](#)

security cookie Spanish text placeholder: [18.3 on page 343.](#) [926](#)

stdout standard output. [21](#)

thunk function Spanish text placeholder. [977, 990](#)

tracer Spanish text placeholder: [70.3 on page 943.](#) [227, 920, 933, 1068](#)

Windows NT Windows NT, 2000, XP, Vista, 7, 8. [776, 938, 1094, 1240](#)

xoring Spanish text placeholder. [926, 998](#)

Índice alfabético

- .NET, 906
- Sintaxis AT&T, 13, 38
- Buffer Overflow, 334, 926
- Elementos del lenguaje C
 - Apuntaores, 77, 84, 130, 504, 550
 - Post-decremento, 574
 - Post-incremento, 574
 - Pre-decremento, 574
 - Pre-incremento, 574
 - C99, 128
 - bool, 389
 - restrict, 660
 - variable length arrays, 348
 - const, 11, 94
 - for, 219, 610
 - if, 146, 184
 - return, 11, 100, 127
 - switch, 182, 184, 192
 - while, 245
- Librería estándar C
 - alloc(), 36, 348, 588, 914
 - assert(), 354, 856
 - atexit(), 731
 - atoi(), 639, 1051
 - calloc(), 1026
 - close(), 892
 - exit(), 592
 - free(), 588
 - fseek(), 1026
 - ftell(), 1026
 - getenv(), 1053
 - localtime(), 789
 - localtime_r(), 461
 - longjmp(), 184
 - malloc(), 451, 588
 - memchr(), 1235
 - memcmp(), 658, 858, 1237
 - memcpy(), 13, 77, 654, 1234
 - memset(), 653, 1083, 1236
 - open(), 892
 - pow(), 282
 - puts(), 21
 - qsort(), 504
 - rand(), 439, 847, 955, 958, 1006
 - read(), 892
 - realloc(), 588
 - scanf(), 77
 - strcmp(), 649, 892
 - strcpy(), 13, 652, 1007
 - strlen(), 245, 546, 652, 673, 1235
 - strstr(), 1268
 - time(), 789
 - tolower(), 1034
 - toupper(), 683
 - va_arg, 665
 - va_list, 669
 - vprintf, 669
- Anomalías del compilador, 174, 282, 367, 401, 422, 623, 680, 1146
- C++, 1070
 - , 914
 - C++11, 746, 879
 - ostream, 721
 - References, 723

RTTI, 721	1082, 1084, 1123, 1134, 1146,
STL, 841	1161
std::forward_list, 746	
std::list, 732	
std::map, 756	
std::set, 756	
std::string, 723	
std::vector, 746	
, 821	
Uso de grep, 227, 319, 841, 859, 864,	
1067	
, 590	
Sintaxis Intel, 13, 17	
Mac OS X, 945	
código independiente de la posición, 18,	
888	
RAM, 94	
ROM, 94	
, 900	
, 88	
, 756	
, 22	
, 94, 702	
, 159	
(NaNs), 311	
, 702	
, 323	
, 341	
, 1110	
Recursión, 32, 34, 606	
Tail recursion, 606	
Pila, 33, 114, 184	
, 34	
, 78	
Azúcar sintáctica, 184	
Modo Thumb-2, 22	
iPod/iPhone/iPad, 17	
OllyDbg, 48, 80, 91, 115, 131, 149, 199,	
224, 248, 273, 290, 301, 328,	
337, 340, 357, 413, 448, 470,	
471, 477, 481, 508, 904, 943,	
1256	
Oracle RDBMS, 11, 533, 855, 910, 1071,	
Ada, 125	
Alpha AXP, 6	
AMD, 875	
Angry Birds, 318, 319	
ARM, 253, 692, 976, 1247	
Modo ARM, 5	
Instrucciones	
ADC, 523	
ADD, 21, 124, 158, 228, 409, 423,	
633, 1249	
ADDAL, 158	
ADDC, 205	
ADDS, 122, 523, 1249	
ADR, 18, 158	
ADRcc, 158, 192, 589	
ADRP/ADD pair, 24, 61, 95, 352,	
368, 577	
ANDcc, 685	
ASR, 427	
ASRS, 401, 633	
B, 60, 158, 159	
Bcc, 111, 112, 174	
BCS, 159, 320	
BEQ, 110, 192	
BGE, 159	
BIC, 401, 402, 407, 429	
BL, 18, 20, 22, 24, 158, 579	
BLcc, 158	
BLE, 159	
BLS, 159	
BLT, 228	
BLX, 22	
BNE, 159	
BX, 121, 207	
CLZ, 1188, 1189	
CMP, 110, 111, 158, 192, 205, 228,	
423, 1249	
CSEL, 171, 177, 180, 424	
EOR, 407	
FCMPE, 321	
FCSEL, 321	

- FMOV, 577
 FMRS, 409
 IT, 180, 318, 347
 LDMccFD, 158
 LDMEA, 33
 LDMED, 33
 LDMFA, 33
 LDMFD, 19, 33, 158
 LDP, 25
 LDR, 62, 84, 94, 330, 351, 574
 LDR.W, 364
 LDRB, 472
 LDRB.W, 253
 LDRSB, 253
 LSL, 423, 427
 LSL.W, 423
 LSLR, 686
 LSLS, 331, 408, 686
 LSR, 427
 LSRS, 408
 MADD, 122
 MLA, 121, 122
 MOV, 8, 18, 20, 423, 633
 MOVcc, 174, 180
 MOVK, 576
 MOVT, 20, 633
 MOVT.W, 22
 MOVW, 22
 MUL, 124
 MULS, 122
 MVNS, 253
 NEG, 647
 ORR, 401
 POP, 17, 19, 20, 33, 35
 PUSH, 20, 33, 35
 RET, 25
 RSB, 166, 364, 423, 647
 SBC, 523
 SMMUL, 633
 STMEA, 33
 STMED, 33
 STMFA, 33, 64
 STMFD, 17, 33
 STMIA, 62
 STMIB, 64
 STP, 23, 61
 STR, 62, 330
 SUB, 62, 364, 423
 SUBcc, 685
 SUBEQ, 254
 SUBS, 523
 SXTB, 472
 SXTW, 368
 TEST, 246
 TST, 393, 423
 VADD, 278
 VDIV, 278
 VLDR, 278
 VMOV, 278, 318
 VMOVG, 318
 VMRS, 318
 VMUL, 278
 XOR, 167, 409
 , 205
 , 121, 207
 Modos de direccionamiento, 574
 , 22
 Registros
 APSR, 318
 FPSCR, 318
 Link Register, 18, 19, 35, 60, 207,
 1248
 R0, 1248
 scratch registers, 253, 1248
 X0, 1249
 Z, 110, 1248
 Modo Thumb, 5, 159, 207
 Modo Thumb-2, 5, 207, 318, 319
 armel, 279
 armhf, 279
 Condition codes, 158
 D-registros, 278, 1248
 Data processing instructions, 633
 DCB, 19
 hard float, 279
 if-then block, 318

- Leaf function, 35
- Optional operators
 - ASR, 423, 633
 - LSL, 330, 364, 423, 576
 - LSR, 423, 633
 - ROR, 423
 - RRX, 423
- S-registros, 278, 1248
- soft float, 279
- ARM64
 - lo12, 61
- ASLR, 901
- AWK, 862
- Base64, 855
- base64, 855
- bash, 127
- BASIC
 - POKE, 866
- binary grep, 858, 947
- BIND.EXE, 906
- binutils, 499
- Bitcoin, 1148
- Borland C++Builder, 843
- Borland Delphi, 843, 851, 1144
- BSoD, 886
- BSS, 902
- C11, 879
- Callbacks, 504
- Canary, 343
- cdecl, 47, 869
- COFF, 986
- column-major order, 356
- Compiler intrinsic, 37, 1145
- CRC32, 590, 607
- CRT, 895, 920
- Cygwin, 843, 848, 907, 945
- DES, 533, 550
- dlopen(), 892
- dlsym(), 892
- DOSBox, 1094
- DosBox, 864
- double, 270, 875
- dtruss, 945
- Duff's device, 628
- EICAR, 1089
- ELF, 92
- Error messages, 855
- fastcall, 14, 76, 391, 870
- float, 270, 875
- Forth, 819
- FORTRAN, 356, 660, 772, 843
- Function epilogue, 32, 60, 62, 158, 472, 862
- Function prologue, 12, 32, 35, 62, 343, 862
- Fused multiply-add, 121, 122
- Fuzzing, 647
- GCC, 842, 1253, 1257
- GDB, 29, 52, 56, 342, 514, 515, 943, 1257
- Glibc, 514, 886
- Hex-Rays, 966
- Hiew, 107, 155, 850, 903, 904, 907, 1144
- IDA, 101, 499, 659, 833, 853, 1130, 1255
 - var_?, 62, 84
- IEEE 754, 270, 403, 494, 560, 1221
- Inline code, 229, 401, 648, 710, 751
- Integer overflow, 125
- Intel
 - 8080, 253
 - 8086, 253, 400, 1001
 - , 787, 1160
 - 8253, 1092
 - 80286, 1001, 1160
 - 80386, 400, 1160
 - 80486, 270
 - FPU, 270
- Intel C++, 11, 533, 1146, 1161, 1233
- Itanium, 1156
- Java, 793
- jumptable, 198, 207

- Keil, 17
 kernel panic, 886
 kernel space, 886
- LD_PRELOAD, 891
 Linux, 392, 888, 1071
 libc.so.6, 391, 513
- LLVM, 17
 long double, 270
 Loop unwinding, 222
- Mac OS Classic, 976
 MD5, 590, 857
 MFC, 904, 1053
 MIDI, 858
 MinGW, 843
 minifloat, 577
 MIPS, 6, 694, 865, 902, 976
 Puntero Global, 25
- Instrucciones
 ADD, 125
 ADD.D, 282
 ADDIU, 26, 98, 99
 ADDU, 125
 AND, 403
 BC1F, 323
 BC1T, 323
 BEQ, 112, 161
 BLTZ, 167
 BNE, 161
 BNEZ, 209
 BREAK, 634
 C.LT.D, 323
 DIV.D, 282
 J, 8, 26
 JAL, 125
 JALR, 26, 125
 JR, 196
 LB, 236
 LBU, 236
 LI, 580
 LUI, 26, 98, 99, 282, 407, 580
 LW, 26, 85, 99, 196
 LWC1, 282
- MFC1, 287
 MFHI, 125, 634, 1252
 MFLO, 125, 634, 1252
 MTC1, 502
 MUL.D, 282
 MULT, 125
 NOR, 256
 OR, 29
 ORI, 403, 580
 SB, 236
 SLL, 209, 262, 427
 SLLV, 427
 SLT, 161
 SLTIU, 209
 SLTU, 161, 163, 209
 SRL, 268
 SUBU, 167
 SW, 70
- Pseudo-instrucciones
 B, 232
 BEQZ, 163
 L.D, 282
 LA, 29
 LI, 8
 MOVE, 26, 97
 NEGU, 167
 NOP, 29, 97
 NOT, 256
- Registros
 FCCR, 322
 HI, 634
 LO, 634
- Branch delay slot, 9
 Global Pointer, 364
 Load delay slot, 196
 O32, 70, 76, 1251
- MS-DOS, 345, 782, 858, 864, 866, 900,
 1001, 1089, 1091, 1116, 1144,
 1160, 1221, 1234, 1240, 1241
 DOS extenders, 1160
- MSVC, 1254, 1257
- Name mangling, 702
 Native API, 901

- NEC V20, 1093
 objdump, 499, 890, 907
 OEP, 899, 906
 opaque predicate, 698
 OpenMP, 846, 1148
 OpenWatcom, 843, 871
 Page (memory), 548
 Pascal, 851
 PDP-11, 574
 PowerPC, 6, 25, 976
 puts() printf(), 83, 126, 156
 puts() en lugar de printf(), 21
 Quake III Arena, 503
 Raspberry Pi, 17
 ReactOS, 917
 Register allocation, 550
 Relocation, 22
 row-major order, 356
 RVA, 900
 SAP, 841, 1065
 SCO OpenServer, 986
 Scratch space, 873
 Security cookie, 343, 926
 Security through obscurity, 855
 SHA1, 590
 SHA512, 1148
 Shadow space, 118, 120, 561
 Shellcode, 697, 886, 901, 1090, 1244
 Signed numbers, 148, 584
 SIMD, 560, 657
 SSE, 560
 SSE2, 560
 stdcall, 869, 1144
 strace, 891, 945
 syscall, 391, 886, 945
 TCP/IP, 587
 thiscall, 702, 704, 872
 thunk-, 22, 905, 977, 989
 TLS, 345, 879, 902, 906, 1225
 , 883
 Callbacks, 906
 tracer, 224, 510, 512, 848, 859, 864, 920, 933, 943, 1068, 1076, 1083, 1086, 1144, 1215
 Unicode, 851
 Unrolled loop, 229, 347, 653
 uptime, 891
 USB, 979
 user space, 886
 UTF-16LE, 851, 852
 UTF-8, 851
 VA, 900
 Watcom, 843
 Windows, 938
 API, 1221
 IAT, 900
 INT, 900
 KERNEL32.DLL, 390
 MSVCR80.DLL, 506
 NTAPI, 950
 ntoskrnl.exe, 1071
 PDB, 841, 903, 950, 958, 1065
 Structured Exception Handling, 38, 907
 TIB, 345, 907, 1225
 Win32, 389, 852, 891, 899, 1160
 GetProcAddress, 906
 LoadLibrary, 906
 Ordinal, 904
 RaiseException(), 907
 SetUnhandledExceptionFilter(), 910
 Windows 2000, 902
 Windows 3.x, 776, 1160
 Windows NT4, 902
 Windows Vista, 900, 950
 Windows XP, 902, 906, 958
 Wine, 917
 Wolfram Mathematica, 636, 638, 968, 1110
 x86

Flags	FABS, 1242
CF, 1231, 1234, 1237, 1241, 1242	FADD, 1242
DF, 1237, 1242	FADDP, 272, 278, 1242
IF, 1237, 1242	FATRET, 421, 422
Instrucciones	FCHS, 1243
AAA, 1246	FCMOVcc, 313
AAS, 1246	FCOM, 300, 311, 1243
ADC, 522, 782, 1231	FCOMP, 288, 1243
ADD, 11, 47, 115, 640, 782, 1231	FCOMPP, 1243
ADDSD, 561	FDIV, 272, 859, 1243
ADDSS, 573	FDIVP, 272, 1243
ADRcc, 169	FDIVR, 278, 1243
AND, 12, 390, 395, 411, 428, 480, 1231, 1236	FDIVRP, 1243
BSF, 548, 1186, 1236	FILD, 1243
BSR, 1236	FIST, 1243
BSWAP, 587, 1236	FISTP, 1243
BT, 1236	FLD, 283, 288, 1243
BTC, 406, 1236	FLD1, 1243
BTR, 406, 940, 1236	FLDCW, 1243
BTS, 406, 1236	FLDZ, 1243
CALL, 11, 34, 689, 905, 1231	FMUL, 272, 1243
CBW, 585, 1237	FMULP, 1243
CDQ, 532, 585, 1237	FNSTCW, 1243
CDQE, 585, 1237	FNSTSW, 288, 311, 1244
CLD, 1237	FSINCOS, 1243
CLI, 1237	FSQRT, 1243
CMC, 1237	FST, 1244
CMOVcc, 159, 169, 172, 175, 180, 589, 1237	FSTCW, 1243
CMP, 100, 1231, 1246	FSTP, 283, 1244
CMPSB, 858, 1237	FSTSW, 1244
CMPSD, 1237	FSUB, 1244
CMPSQ, 1237	FSUBP, 1244
CMPSW, 1237	FSUBR, 1244
COMISD, 570	FSUBRP, 1244
COMISS, 573	FUCOM, 311, 1244
CPUID, 477, 1240	FUCOMI, 313
CWD, 585, 782, 1104, 1237	FUCOMP, 1244
CWDE, 585, 1237	FUCOMPP, 311, 1244
DEC, 247, 1231, 1246	FWAIT, 270
DIV, 585, 1240	FXCH, 1244
DIVSD, 561, 861	IDIV, 585, 631, 1240
	IMUL, 115, 367, 585, 1231, 1246
	IN, 689, 1001, 1092, 1240

INC, 247, 1144, 1231, 1246
 INT, 1090, 1240
 INT3, 848
 IRET, 1240
 JA, 148, 312, 584, 1231, 1246
 JAE, 148, 1231, 1246
 JB, 148, 584, 1231, 1246
 JBE, 148, 1231, 1246
 JC, 1231
 Jcc, 112, 174
 JCXZ, 1231
 JE, 184, 1231, 1246
 JECXZ, 1231
 JG, 148, 584, 1231
 JGE, 147, 1231
 JL, 148, 584, 1231
 JLE, 147, 1231
 JMP, 34, 60, 905, 1144, 1231
 JNA, 1231
 JNAE, 1231
 JNB, 1231
 JNBE, 312, 1231
 JNC, 1231
 JNE, 100, 147, 1231, 1246
 JNG, 1231
 JNGE, 1231
 JNL, 1231
 JNLE, 1231
 JNO, 1231, 1246
 JNS, 1231, 1246
 JNZ, 1231
 JO, 1231, 1246
 JP, 288, 1093, 1231, 1246
 JPO, 1231
 JRCXZ, 1231
 JS, 1231, 1246
 JZ, 110, 184, 1146, 1231
 LAHF, 1232
 LEA, 79, 118, 454, 594, 612, 640,
 874, 953, 1233
 LEAVE, 12, 1233
 LES, 1008, 1103
 LOCK, 939
 LODSB, 1093
 LOOP, 219, 238, 862, 1104, 1240
 MAXSD, 570
 MOV, 8, 11, 14, 653, 654, 689, 904,
 1144, 1234
 MOVDQA, 537
 MOVDQU, 537
 MOVSБ, 1234
 MOVSD, 568, 656, 1033, 1234
 MOVSDX, 568
 MOVSQ, 1234
 MOVSS, 573
 MOVSW, 1234
 MOVSX, 246, 253, 470, 472, 585,
 1234
 MOVSXD, 349
 MOVZX, 247, 451, 976, 1234
 MUL, 585, 1234
 MULSD, 561
 NEG, 645, 1234
 NOP, 612, 1141, 1144, 1234
 NOT, 251, 253, 1038, 1234
 OR, 395, 673, 1234
 OUT, 689, 1001, 1241
 PADDD, 537
 PCMPEQB, 548
 PLMULHW, 533
 PLMULLD, 533
 PMOVMSKB, 548
 POP, 11, 33, 34, 1234, 1246
 POPA, 1241, 1246
 POPCNT, 1241
 POPF, 1092, 1241
 PUSH, 11, 12, 33, 34, 78, 689, 1234,
 1246
 PUSHA, 1241, 1246
 PUSHF, 1241
 PXOR, 548
 RCL, 862, 1241
 RCR, 1241
 RET, 7, 11, 34, 343, 704, 777, 1144,
 1234
 ROL, 421, 1145, 1241

- ROR, [1145](#), [1241](#)
 SAHF, [311](#), [1234](#)
 SAL, [1242](#)
 SALC, [1093](#)
 SAR, [427](#), [585](#), [663](#), [1104](#), [1242](#)
 SBB, [522](#), [1234](#)
 SCASB, [1093](#), [1235](#)
 SCASD, [1235](#)
 SCASQ, [1235](#)
 SCASW, [1235](#)
 SETALC, [1093](#)
 SETcc, [161](#), [247](#), [312](#), [1242](#)
 SHL, [261](#), [327](#), [427](#), [1235](#)
 SHR, [267](#), [427](#), [480](#), [1235](#)
 SHRD, [530](#), [1236](#)
 STC, [1242](#)
 STD, [1242](#)
 STI, [1242](#)
 STOSB, [1236](#)
 STOSD, [1236](#)
 STOSQ, [654](#), [1236](#)
 STOSW, [1236](#)
 SUB, [11](#), [12](#), [100](#), [184](#), [640](#), [1231](#),
 [1236](#)
 SYSCALL, [1240](#), [1242](#)
 SYSENTER, [887](#), [1240](#), [1242](#)
 TEST, [246](#), [390](#), [393](#), [428](#), [1236](#)
 UD2, [1242](#)
 XADD, [940](#)
 XCHG, [1234](#), [1236](#)
 XOR, [11](#), [100](#), [251](#), [663](#), [862](#), [993](#),
 [1144](#), [1236](#), [1246](#)
- Prefijos
 LOCK, [940](#), [1231](#)
 REP, [1231](#), [1234](#), [1236](#)
 REPE/REPNE, [1231](#)
 REPNE, [1235](#)
- Registros
 Flags, [100](#), [149](#), [1225](#)
 AH, [1232](#), [1234](#)
 CS, [1160](#)
 DR6, [1229](#)
 DR7, [1229](#)

Bibliografía

- [al12] Nick Montfort et al. 10 PRINT CHR\$(205.5+RND(1)); : GOTO 10. También disponible como <http://go.yurichev.com/17286>. The MIT Press, 2012.
- [AMD13a] AMD. AMD64 Architecture Programmer's Manual. También disponible como <http://go.yurichev.com/17284>. 2013.
- [AMD13b] AMD. Software Optimization Guide for AMD Family 16h Processors. También disponible como <http://go.yurichev.com/17285>. 2013.
- [App10] Apple. iOS ABI Function Call Guide. También disponible como <http://go.yurichev.com/17276>. 2010.
- [ARM13a] ARM. ARM Architecture Reference Manual, ARMv8, for ARMv8-A architecture p 2013.
- [ARM13b] ARM. ELF for the ARM 64-bit Architecture (AArch64). También disponible como <http://go.yurichev.com/17288>. 2013.
- [ARM13c] ARM. Procedure Call Standard for the ARM 64-bit Architecture (AArch64). También disponible como <http://go.yurichev.com/17287>. 2013.
- [Bro] Ralf Brown. The x86 Interrupt List. También disponible como <http://go.yurichev.com/17292>.
- [Bur] Mike Burrell. «Writing Efficient Itanium 2 Assembly Code». En: (). También disponible como <http://go.yurichev.com/17265>.
- [Cli] Marshall Cline. C++ FAQ. También disponible como <http://go.yurichev.com/17291>.
- [Cor+09] Thomas H. Cormen et al. Introduction to Algorithms, Third Edition. 3rd. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.
- [Dij68] Edsger W. Dijkstra. «Letters to the editor: go to statement considered harmful». En: Commun. ACM 11.3 (mar. de 1968), págs. 147-148. ISSN: 0001-0782. DOI: [10.1145/362929.362947](https://doi.org/10.1145/362929.362947). URL: <http://go.yurichev.com/17299>.

- [Dol13] Stephen Dolan. «mov is Turing-complete». En: (2013). También disponible como <http://go.yurichev.com/17269>.
- [Dre07] Ulrich Drepper. What Every Programmer Should Know About Memory. También disponible como <http://go.yurichev.com/17341>. 2007.
- [Dre13] Ulrich Drepper. «ELF Handling For Thread-Local Storage». En: (2013). También disponible como <http://go.yurichev.com/17272>.
- [Fog13a] Agner Fog. Optimizing software in C++: An optimization guide for Windows, Linux and Mac OS X. <http://go.yurichev.com/17279>. 2013.
- [Fog13b] Agner Fog. The microarchitecture of Intel, AMD and VIA CPUs / An optimization guide for Windows, Linux and Mac OS X. <http://go.yurichev.com/17278>. 2013.
- [haq] papasutra of haquebright. «WRITING SHELLCODE FOR IA-64». En: (). También disponible como <http://go.yurichev.com/17340>.
- [IBM00] IBM. PowerPC(tm) Microprocessor Family: The Programming Environments for the PowerPC. También disponible como <http://go.yurichev.com/17281>. 2000.
- [Int13] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Edition. También disponible como <http://go.yurichev.com/17283>. 2013.
- [Int14] Intel. Intel® 64 and IA-32 Architectures Optimization Reference Manual. También disponible como <http://go.yurichev.com/17342>. September 2014.
- [ISO07] ISO. ISO/IEC 9899:TC3 (C C99 standard). También disponible como <http://go.yurichev.com/17274>. 2007.
- [ISO13] ISO. ISO/IEC 14882:2011 (C++ 11 standard). También disponible como <http://go.yurichev.com/17275>. 2013.
- [Jav13] Java. The Java® Virtual Machine Specification Java SE 7 Edition. También disponible como <http://go.yurichev.com/17345> y <http://go.yurichev.com/17346>. February 2013.
- [Ker88] Brian W. Kernighan. The C Programming Language. Ed. por Dennis M. Ritchie. 2nd. Prentice Hall Professional Technical Reference, 1988. ISBN: 0131103709.
- [Knu74] Donald E. Knuth. «Structured Programming with go to Statements». En: ACM Comput. Surv. 6.4 (dic. de 1974). Also available as <http://go.yurichev.com/17271>, págs. 261-301. ISSN: 0360-0300. DOI: [10.1145/356635.356640](https://doi.org/10.1145/356635.356640). URL: <http://go.yurichev.com/17300>.
- [Loh10] Eugene Loh. «The Ideal HPC Programming Language». En: Queue 8.6 (jun. de 2010), 30:30-30:38. ISSN: 1542-7730. DOI: [10.1145/1810226.1820518](https://doi.org/10.1145/1810226.1820518). URL: <http://go.yurichev.com/17298>.

- [Ltd94] Advanced RISC Machines Ltd. The ARM Cookbook. También disponible como <http://go.yurichev.com/17273>. 1994.
- [Mit13] Michael Matz / Jan Hubicka / Andreas Jaeger / Mark Mitchell. System V Application Interface. También disponible como <http://go.yurichev.com/17295>. 2013.
- [Pie] Matt Pietrek. «A Crash Course on the Depths of Win32™ Structured Exception Handling». En: MSDN magazine (). URL: <http://go.yurichev.com/17293>.
- [Pie02] Matt Pietrek. «An In-Depth Look into the Win32 Portable Executable File Format». En: MSDN magazine (2002). URL: <http://go.yurichev.com/17318>.
- [Pre+07] William H. Press et al. Numerical Recipes. 2007.
- [RA09] Mark E. Russinovich y David A. Solomon with Alex Ionescu. Windows® Internals. 2009.
- [Ray03] Eric S. Raymond. The Art of UNIX Programming. También disponible como <http://go.yurichev.com/17277>. Pearson Education, 2003. ISBN: 0131429019.
- [Rit79] Dennis M. Ritchie. «The Evolution of the Unix Time-sharing System». En: (1979).
- [Rit86] Dennis M. Ritchie. Where did ++ come from? (net.lang.c). <http://go.yurichev.com/17296>. [Online; accessed 2013]. 1986.
- [Rit93] Dennis M. Ritchie. «The development of the C language». En: SIGPLAN Not. 28.3 (mar. de 1993). También disponible como <http://go.yurichev.com/17264>, págs. 201-208. ISSN: 0362-1340. DOI: [10.1145/155360.155580](https://doi.org/10.1145/155360.155580). URL: <http://go.yurichev.com/17297>.
- [RT74] D. M. Ritchie y K. Thompson. «The UNIX Time Sharing System». En: (1974). También disponible como <http://go.yurichev.com/17270>.
- [Sch94] Bruce Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 1994.
- [SK95] SunSoft Steve Zucker e IBM Kari Karhi. SYSTEM V APPLICATION BINARY INTERFACE. También disponible como <http://go.yurichev.com/17282>. 1995.
- [Sko12] Igor Skochinsky. Compiler Internals: Exceptions and RTTI. También disponible como <http://go.yurichev.com/17294>. 2012.
- [Swe10] Dominic Sweetman. See MIPS Run, Second Edition. 2010.
- [War02] Henry S. Warren. Hacker's Delight. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0201914654.

- [Yur12] Dennis Yurichev. «Finding unknown algorithm using only input/output pairs and Z3 SMT solver». En: (2012). También disponible como <http://go.yurichev.com/17268>.
- [Yur13] Dennis Yurichev. C/C++ programming language notes. También disponible como <http://go.yurichev.com/17289>. 2013.