Assembly-Level Debugging

Alexandro Sánchez Bach



50 Years – Research for A Life Without Cancer



3/18/2015 | Page 2



- Issue found in a multi-threaded algorithm.
- Random worker threads crashing randomly at completely unrelated points.

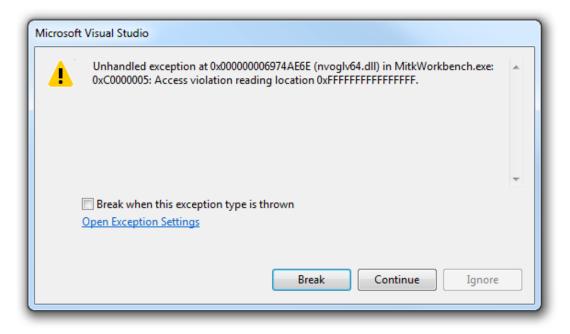
ne la	Language 🔶	Name	Language 🔶
I.dll!000000077b79997()	Unknown	sysfer.dll!00000007563af22()	Unknown
I.dll!000000077b76a1f()	Unknown	ntdll.dll!000000077b5b8b4()	Unknown
I.dll!000000077b76978()	Unknown	ntdll.dll!000000077b76aab()	Unknown
nel32.dll!000000077a259f5()	Unknown	ntdll.dll!000000077b76978()	Unknown
I.dll!000000077b5c541()	Unknown	kernel32.dll!000000077a259f5()	Unknown
		ntdll.dll!000000077b5c541()	Unknown
Call Stack		₽ × Call Stack	
Name	Language	 Name 	Langua
ntdll.dll!000000077bf40cf()	Unknown	ntdll.dll!000000077bf40cf()	Unknov
ntdll.dll!000000077bf4746()	Unknown	ntdll.dll!000000077bf4746()	Unknov
ntdll.dll!000000077bf5952()	Unknown	ntdll.dll!000000077bf5952()	Unknov
ntdll.dll!000000077bf7604()	Unknown	ntdll.dll!000000077bf7604()	Unknov
ntdll.dll!000000077b8fb2a()	Unknown	ntdll.dll!000000077b914e5()	Unknov
ntdll.dll!000000077b834d8()	Unknown	ntdll.dll!000000077b840fd()	Unknov
ntdll.dll!000000077c070dd()	Unknown	ntdll.dll!000000077bf9be9()	Unknov
ntdll.dll!000000077bcb5aa()	Unknown	ntdll.dll!000000077b9db50()	Unknov
ntdll.dll!000000077b834d8()	Unknown	ntdll.dll!000000077b840fd()	Unknov
msvcr110d.dll!_heap_alloc_base(unsignedint64 size)	Line 57 C	kernel32.dll!000000077a31a4a()	Unknov
msvcr110d.dll!_heap_alloc_dbg_impl(unsignedint64	nSize, int nBlockUse, const char * szFile C++	nvoglv64.dll!000000069c941c4()	Unknov
msvcr110d.dll!_nh_malloc_dbg_impl(unsignedint64	nSize, int nhFlag, int nBlockUse, const (C++	nvoglv64.dll!000000069c9af80()	Unknov
msvcr110d.dll!_calloc_dbg_impl(unsignedint64 nNu	m, unsignedint64 nSize, int nBlockUs C++	ntdll.dll!000000077b5b8b4()	Unknov
msvcr110d.dll!_calloc_dbg(unsignedint64 nNum, un	signedint64 nSize, int nBlockUse, cor C++	ntdll.dll!000000077b76aab()	Unknov
msvcr110d.dll!CRTDLL_INIT(void * hDllHandle, unsig	ned long dwReason, void * lpreserved) C	ntdll.dll!000000077b76978()	Unknov
msvcr110d.dll!_CRTDLL_INIT(void * hDllHandle, unsign	ed long dwReason, void * lpreserved) l C	kernel32.dll!000000077a259f5()	Unknov
ntdll.dll!000000077b5c78c()	Unknown	ntdll.dll!000000077b5c541()	Unknov
ntdll.dll!000000077b5c44f()	Unknown		
ntdll.dll!000000077b5c34e()	Unknown		



3/18/2015 | Page 3



• Our IDE is not helping much either.



- No source code context.
- No time/motivation to make wild guesses about what happened.





- Known in Visual Studio as Address-Level Debugging, and provide access to:
 - Memory windows
 - Disassembly panel
 - CPU registers panel
 - Memory write/execute breakpoints
- Additionally, other solutions (e.g. *OllyDbg*) provide breakpoints on memory reading attempts as well.
- Alternatives (discussed later):
 - GDB
 - Xcode

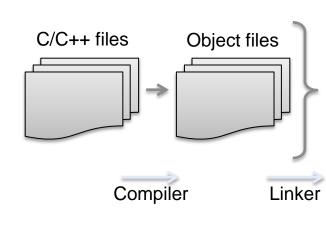
Computer Assisted Interventions (E131)

3/18/2015 | Page 5





50 Years – Research for A Life Without Cancer

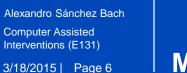


	• • •	••••
0000000077B833E9	shr	rcx,4
0000000077B833ED	cmp	rcx, rax
0000000077B833F0	jae	0000000077B84B29
0000000077B833F6	mov	r8d, dword ptr [rd>
0000000077B833FA	dec	r8d
0000000077B833FD	cmp	rcx,r8
0000000077B83400	jae	0000000077B834AA
0000000077B83406	mov	eax,dword ptr [rd>
0000000077B83409	sub	rcx, rax
0000000077B8340C	cmp	dword ptr [rdx+0Ch
000000077B8340F	je	0000000077B83414
0000000077B83411	xor	rax, rax
0000000077B83414	mov	qword ptr [rcx + 8
0000000077B83418 0000000077B8341C	lea	rbx, [rax+rcx*8]
0000000077B8341C	test je	rbx,rbx 0000000077B834B3
00000000077B8341F	mov	
00000000077B83429	test	rcx, qword ptr [rb; cl,1
00000000077B83429	je	0000000077B834B3
00000000077B83432	dec	rcx
00000000077B83435	mov	rdx, rsi
00000000077B83438	mov	dword ptr [rsp+110
00000000077B83443	call	0000000077B84ED0
00000000077B83448	test	rax, rax
00000000077B8344B	je	00000000077B834B3
00000000077B8344D	mov	rdi,rax
0000000077B83450	test	bpl,8
0000000077B83454	ine	0000000077B85921
0000000077B8345A	test	rdi,rdi
0000000077B8345D	ie	0000000077BCBACC
0000000077B83463	test	r14d,r14d
0000000077B83466	jne	0000000077BCBBC2
0000000077B8346C	cmp	byte ptr [7FFE0380
0000000077B83474	mov	r15,qword ptr [rsp
0000000077B8347C	mov	r14, gword ptr [rsp
0000000077B83484	mov	r13,qword ptr [rsp
0000000077B8348C	mov	rbx,qword ptr [rsp
0000000077B83494	jne	0000000077BCBC32
0000000077B8349A	mov	rax,rdi
0000000077B8349D	add	rsp,0E8h
0000000077B834A4	pop	r12
0000000077B834A6	pop	rdi
0000000077B834A7	pop	rsi
0000000077B834A8	pop	rbp

Disassembly

9 rdx+8]		rax, rax
	mov	qword ptr [rcx + 8], rax
A rdx+18h]	_	
OCh],ebx 4		Registers
+ 8],rax		
		General Purpose registers:
3 rbx+8]		RAX, RBX, RCX, RDX, RSP, RBP, RSI,
3		RDI, R8, R9, R10, R11, R12, R13, R14, R15
110h],2 0		FPU/SSE/AVX extension registers
3		Segment registers
1		
c		
2 380h] , 0		Memory
rsp+0D0h] rsp+0D8h] rsp+0E0h] rsp+118h]		 0x00000000699F720 1b 00 10 00 80 1f 00 00€ 0x00000000699F728 33 00 00 00 00 00 00 00 3
2		0x00000000699F730 00 00 2b 00 00 02 00 00+ 0x0000000699F738 00 00 00 00 00 00 00 00
		0x00000000699F748 e0 44 c3 77 00 00 00 00 àDÃw

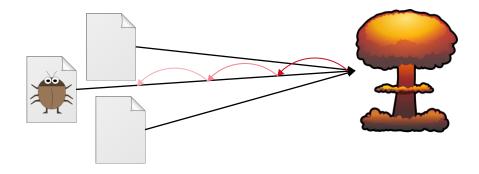
0x00000000699F720	1b 00	10 00	0 80 3	1f 00	00	€
0x00000000699F728	33 00	00 00	0 00 0	00 00	00	3
0x00000000699F730	00 00	2b 00	0 00 0	02 00	00	+
0x00000000699F738	00 00	00 00	0 00 0	00 00	00	
0x00000000699F748	e0 44	c3 7'	7 00 0	00 00	00	àDÃw
0x00000000699F750	00 00	00 00	0 00 0	00 00	00	
0x00000000699F758	3b 0f	b5 7'	7 00 0	00 00	00	;.µw
0x00000000699F760	70 57	bf 03	3 00 0	00 02	00	pW;
0x00000000699F768	00 00	00 00	0 00 0	00 01	00	
0x00000000699F778	30 d3	dd 01	3 00 0	00 00	00	0ÓÝ



Motivation



- Spotting bugs where the information provided by the IDE's standard debugging tools is not enough.
- Non-trivial bugs in multi-threaded algorithms.
- "Random" bugs.
- Trashed stack or heap overflow related bugs.
- Tracking back the cause of the bug from its effects.





3/18/2015 | Page 7



• Back to our MITK example.

Example

Microsoft Visual Studio	
Unhandled exception at 0x0000006974AE6E (nvoglv64.dll) in MitkWorkbench.exe: 0xC0000005: Access violation reading location 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	*
Break when this exception type is thrown Open Exception Settings	Ŧ
Break Continue Ignore	



3/18/2015 | Page 8



dkfz.

• Current instruction in the disassembly panel.

A	ddress: 000000007563a	af22()	
I	Viewing Options		
	000000007563AF1A	push	rdi
	000000007563AF1B	sub	rsp,20h
	000000007563AF1F	mov	rbx,rcx
⇔	000000007563AF22	mov	rcx,qword ptr [rcx+38h]
	000000007563AF26	test	rcx,rcx
	000000007563AF29	je	000000007563AF30

Current state of the thread.

		RBX	=	4376000030484843	DOM				
000000000				4371 000030A0A0A3	RCX	=	437F00003DA8A8A3		
	000000	RSI	=	000000000000000000000000000000000000000	RDI	=	000000000000000000000000000000000000000		
00000000B	63FAB8	R9	=	000000000000000000000000000000000000000	R10	=	000000000000000000000000000000000000000		
0000000000	2A0DD0	R12	=	000000000A7AD8F0	R13	=	000007FFFFFDF000		
0000000000	000002	R15	=	000000000000000000000000000000000000000	RIP	=	000000007563AF22		
00000000B	63F950	RBP	=	000000000000000000	EFL	=	00010204		
0003da8a8	db = 00	0000	000	00000000					
	000000000 000000000 00000000B	0000000002A0DD0 0000000000000002 000000000B63F950	00000000002A0DD0 R12 000000000000002 R15 000000000863F950 RBP	0000000002A0DD0 R12 = 0000000000000002 R15 = 000000000863F950 RBP =	0000000002A0DD0 R12 = 000000000A7AD8F0 000000000000002 R15 = 00000000000000000	0000000002A0DD0 R12 = 000000000A7AD8F0 R13 0000000000000002 R15 = 000000000000000 RIP 000000000B63F950 RBP = 0000000000000007 EFL	0000000002A0DD0 R12 = 00000000A7AD8F0 R13 = 0000000000000002 R15 = 000000000000000 RIP = 000000000863F950 RBP = 000000000000000 EFL =	0000000002A0DD0 R12 = 00000000A7AD8F0 R13 = 000007FFFFDF000 000000000000002 R15 = 00000000000000 RIP = 000000007563AF22 00000000863F950 RBP = 00000000000000 EFL = 00010204	0000000002A0DD0 R12 = 00000000A7AD8F0 R13 = 000007FFFFDF000 000000000000002 R15 = 00000000000000 RIP = 000000007563AF22 00000000B63F950 RBP = 00000000000007 EFL = 00010204



3/18/2015 | Page 9



50 Years – Research for

A Life Without Cancer

-

• Current instruction in the disassembly panel.

A	ddress: 00)0000007563a	f22()	
I	Viewing	Options		
	0000000	07563AF1A	push	rdi
	0000000	07563AF1B	sub	rsp,20h
	0000000	07563AF1F	mov	rbx,rcx
⇔	0000000	07563AF22	mov	rcx,qword ptr [rcx+38h]
	00000000	07563AF26	test	rcx,rcx
	0000000	07563AF29	je	00000007563AF30

Current state of the thread.

(AX =	00000000	7563AFØC	RBX	= 4	437F00003DA	48A8A3	RCX	=	437F00003DA8A8A3	
DX =	00000000	00000000	RSI	= (0000000000	000010	RDI	=	00000000000000000	
8 =	00000000	0B63FAB8	R9	= (00000000000	000000	R10	=	000000000000000000	
11 =	00000000	002A0DD0	R12	= (00000000A7	7AD8F0	R13	=	000007FFFFFDF000	
14 =	00000000	00000002	R15	= (00000000000	000000	RIP	=	000000007563AF22	
SP =	00000000	0B63F950	RBP	= (000000000000000000000000000000000000000	00007	EFL	=	00010204	
	000003da8;	- 2db - 00		0.01	0000000					
X000	0000050466	400D = 00	00000	000	00000000					





- *Guessing** data types from their raw representation.
 - Floats: ~0x4000000 or ~0xc000000
 - Doubles: ~0x400000000000 or ~0xc000000000000000
 - Addresses: 0x0000xxxxxxxxx or 0xffff0000000000
 - Integers: 0xxxxxxxxxxxxxxxxx
 - Uninit. data**: 0xccccccc (stack) or 0xcDcDcDcD (heap)
- Current state of the thread.

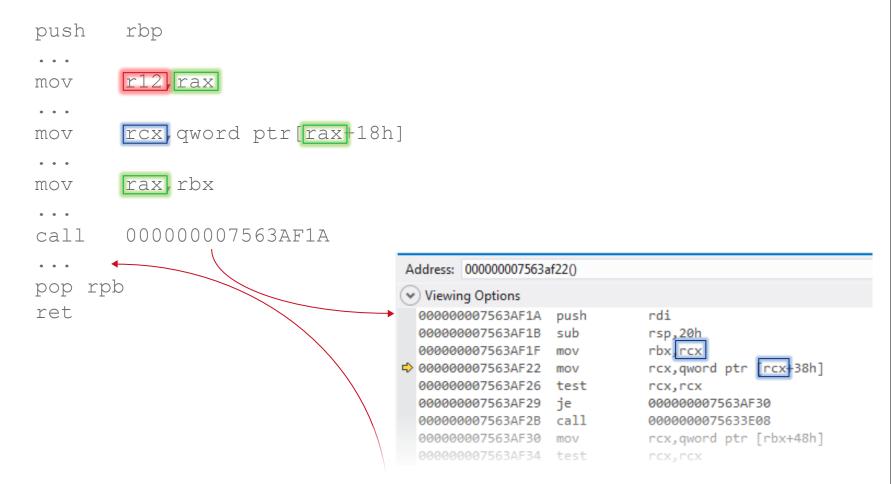
RAX	=	0	000	00	90	75	63	AF	-00		RB)	×	=	43	37	F0	00	003	3D/	48	A8	A3	RC	Х	=	437	FØ(90	930	3AC	BA8	BA3	3				
RDX	=	0	000	00	00	00	00	00	300	3	RS:	Ι	=	00	90	00	00	00	900	<u>9</u> 0	00	10	RD	Ι	=	000	<u>9</u> 06	90	906	900	900	01					
R8	=	0	000	00	00	0B	63	F/	AB8	3	R9		=	00	90	00	00	900	900	0 0	00	00	R1	0	=	000	0 00	900	900	900	900	00)				
R11	=	0	000	00	00	00	2A	0	DDO	3	R1:	2	=	00	90	00	00	000	ðA:	7A	D8	FØ	R1	3	=	000	00 7	7FI	FFF	FC)FØ	00)				
R14	=	0	000	00	00	00	00	00	302	2	R1!	5	=	00	90	00	00	000	900	8 0	00	00	RI	Р	=	000	300	900	075	563	BAF	22	2				
RSP	=	0	000	00	00	0B	63	FS	950	3	RBI	P	=	00	90	00	00	000	900	90	00	07	EF	L	=	000	102	204	1								
0x0(00	90	000	3d	a8	a8	db) =	= (90	00	90	90	00	90(30	00	0																			

- * **Disclaimer**: This is not science, but just a shortcut.
- ** Only on debug configuration. Magic values for Visual Studio only.





• Where did the RCX value come from?







- The base address is still preserved in R12.
- The value that is written into RCX is at the address: R12 + 0x18.

AX	=	000	000	900	756	53A	FØC	R	BX	=	437	/F00	900	3DA	48A	8A3	RCX	=	43	7F000	03D/	48A8	A3		
DX	=	000	000	900	900	000	000	R	SI	=	000	000	000	000	900	010	RDI	=	00	00000	0000	9000	01		
8	=	000	000	900	0B6	i3F/	AB8	R	9	=	000)00(900	000	900	000	R10	=	00	00000	0000	9000	00		
11	=	000	000	900	00 2	2AØ	DD0	R	12	=	000	000	900	0A7	7AD	8F0	R13	=	00	00076	FFF	DFØ	00		
14	=	000	000	900	900	000	002	R	15	=	000	000	900	000	900	000	RIP	=	00	00000	075	53AF	22		
SP	=	000	000	900	0B6	j3F	950	R	BP	=	000	000	900	000	900	007	EFL	=	00	01020)4				
)×00	900	000	030	la8	a8d	lb	= 0	00	000	900	000	000	000												

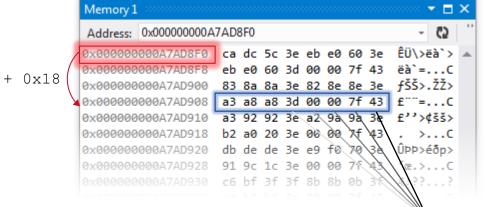


3/18/2015 | Page 13



• Current state of the memory:

Example



Little endianness

Current state of the thread:

							- 1	437F00003DA8A8A3	
					000000000000000000000000000000000000000			000000000000000000000000000000000000000	
								000007FFFFFDF000	
R14 =	0000000	000000002	R15	= "	000000000000000000000000000000000000000	RIP	=	000000007563AF22	
RSP =	0000000	00B63F950	RBP	=	00000000000000007	EFL	=	00010204	
0×000	00003da	8a8db = 0	00000	00	00000000				



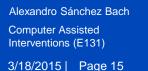
3/18/2015 | Page 14

Example



• Current state of the memory:

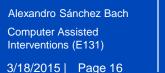
Memory 1			000000 👻 🗖 🗙	
Address: 0x00000000A	7A8AC8		- 🖸 "	
0x000000000A7A8AC8 0x000000000A7A8AD0 0x000000000A7A8AD8 0x000000000A7A8AE0 0x000000000A7A8AE8 0x000000000A7A8AF0 0x00000000A7A8AF8 0x00000000A7A8B00 0x00000000A7A8B08 0x00000000A7A8B18 0x00000000A7A8B18 0x00000000A7A8B20		cd cd cd cd cd cd cd cd cd od cd cd cd cd od od 7f 43 od 00 7f 43	<pre>11111111111 1111111111 1111111111 111111</pre>	Uninitialized heap buffer
				Floating-point data





- If all pointers to the overflown buffer 0x0A7A8B00 had disappeared, there would be a memory leak.
- Therefore, this address should be still accessible inside some function(s) in the call stack of some thread(s).

🖭 Thread:	[2404] Main Thread 🗾 🔻 🔻 Stack Frame: vtkStreamingDemandDriven					
	[2404] Main Thread					
vtkComposite	[1968] ctkEASyncMasterThread					
331	[9072] Thread (pooled)					
332	[11104] ntdll.dll thread					
333	[5056] ntdll.dll thread					
334	[11240] ntdll.dll thread					
335	[8932] RPC Callback Thread					
336	[7084] nvoglv64.dll thread					
337	[5764] nvoglv64.dll thread					
338	[3388] ntdll.dll thread					
339	[864] vtkCommonExecutionModel-6.1d.dll!vtkThreadedImageAlgorithmThreadedExecute					
340	[10372] vtkCommonExecutionModel-6.1d.dll!vtkThreadedImageAlgorithmThreadedExecute					
341	[11040] vtkCommonExecutionModel-6.1d.dll!vtkThreadedImageAlgorithmThreadedExecute					
342	[12972] vtkCommonExecutionModel-6.1d.dll!vtkThreadedImageAlgorithmThreadedExecute					





• Since the address was allocated by our own code, we have debugging symbols and the buffer is shown as:

Value	Туре
0x000000000000000 {}	float *

- We are no longer in a *"source code*"-less situation.
- Allocation happens in a fixed thread.
- Therefore, now we can continue using the standard debugging tools our IDE provides.







- Outsmarting the compiler in hot loops.
- Gaining a better understanding of compilers.

```
if ( !managedapp )
                            dword ptr [managedapp (013F4C66B8h)],0
000000013F4AE4B3 cmp
000000013F4AE4BA ine
                             tmainCRTStartup+1B8h (013F4AE4C8h)
               exit(mainret);
                            ecx, dword ptr [mainret (013F4C6698h)]
00000013F4AE4BC mov
               exit(mainret);
                            qword ptr [ imp exit (013F4CB4A8h)]
000000013F4AE4C2 call
           if (has_cctor == 0)
000000013F4AE4C8 cmp
                            dword ptr [has cctor (013F4C669Ch)],0
                            tmainCRTStartup+1C7h (013F4AE4D7h)
000000013F4AE4CF jne
               cexit();
000000013F4AE4D1 call
                            qword ptr [ imp cexit (013F4CB548h)]
```

- Reverse engineering.
- Fastest understanding of code written by others.



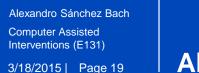
Alexandro Sánchez Bach Computer Assisted Interventions (E131)

3/18/2015 | Page 18



• Any questions?









- All of them are based on the same tools:
 - Memory panels. Mostly focused on:
 - Stack
 - Heap area
 - .bss sections
 - Disassembly panel
 - Registers panel
 - Memory read/write/execute breakpoints
- Alternatives:
 - GDB
 - Xcode

Alexandro Sánchez Bach

Computer Assisted Interventions (E131)

3/18/2015 | Page 20



Alternatives

• GDB

gdb\$ run

[regs]	
EAX: BFFFF5FC EBX: B7FCAFFC ECX: B7FCD19C EDX: 00000001 o d I t S z a P c	
ESI: BFFFF5F4 EDI: BFFFF580 EBP: BFFFF568 ESP: BFFFF550 EIP: 080483AA	
CS: 0073 DS: 007B ES: 007B FS: 0000 GS: 0033 SS: 007B	
[007B:BFFFF550][stack]	
BFFFF5A0 : 00 00 00 00 F8 0F 00 B8 - 01 00 00 00 D0 82 04 08	
BFFFF590 : 70 F5 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 pM	
BFFFF580 : FC AF FC B7 00 00 00 00 - 80 F5 FF BF C8 F5 FF BF	
BFFFF570 : 01 00 00 00 F4 F5 FF BF - FC F5 FF BF 6C 5B FF B71[
BFFFF560 : 00 00 00 00 E0 0C 00 B8 - C8 F5 FF BF 14 4E EB B7N.	
BFFFF550 : FC AF FC B7 FC AF FC B7 - 18 95 04 08 FC AF FC B7	
[007B:BFFFF550][data]	
BFFFF550 : FC AF FC B7 FC AF FC B7 - 18 95 04 08 FC AF FC B7	
BFFFF560 : 00 00 00 00 E0 0C 00 B8 - C8 F5 FF BF 14 4E EB B7N.	
BFFFF570 : 01 00 00 00 F4 F5 FF BF - FC F5 FF BF 6C 5B FF B71[
BFFFF580 : FC AF FC B7 00 00 00 00 - 80 F5 FF BF C8 F5 FF BF	
BFFFF590 : 70 F5 FF BF D2 4D EB B7 - 00 00 00 00 00 00 00 00 pM	
BFFFF5A0 : 00 00 00 00 F8 0F 00 B8 - 01 00 00 00 D0 82 04 08	
BFFFF5B0 : 00 00 00 00 A0 5A FF B7 - B0 66 FF B7 F8 0F 00 B8Zf	
BFFFF5C0 : 01 00 00 00 D0 82 04 08 - 00 00 00 00 F1 82 04 08	
[0073:080483AA][code]	
0x80483aa <main+6>: and esp,0xfffffff0</main+6>	
0x80483ad <main+9>: mov eax,0x0</main+9>	
0x80483b2 <main+14>: add eax,0xf</main+14>	
0x80483b5 <main+17>: add eax,0xf</main+17>	
0x80483b8 <main+20>: shr eax,0x4</main+20>	
0x80483bb <main+23>: shl eax,0x4</main+23>	
0x80483be <main+26>: sub esp,eax</main+26>	
0x80483c0 <main+28>: mov eax,ds:0x80484f4</main+28>	
0x80483c5 <main+33>: mov DWORD PTR [ebp-24],eax</main+33>	
0x80483c8 <main+36>: mov al,ds:0x80484f8</main+36>	
0x80483cd <main+41>: mov BYTE PTR [ebp-20],al</main+41>	
0x80483d0 <main+44>: sub esp,0xc</main+44>	
0x80483d3 <main+47>: push 0x80484f9</main+47>	
0x80483d8 <main+52>: call 0x80482b8 <printf@plt></printf@plt></main+52>	
0x80483dd <main+57>: add esp,0x10</main+57>	
0x80483e0 <main+60>: leave</main+60>	

Alexandro Sánchez Bach

Computer Assisted Interventions (E131)

3/18/2015 | Page 21





Xcode

	A SMS_Demo) 📢 iPhone 5	F	tunning SMS_Demo	o on iPhone 5
		器 < > 圖 s	SMS_Demo > 🧃 Th	ead 1 👌 🔟 0 swift	_dynamicCastObjCClassUnconditional
SMS_Demo	Ø did			namicCastObjCCl	assUnconditional:
PID 940, Paused		<pre>2 0x207f310: 3 0x207f311:</pre>	pushl %ebp movl %esp, %	hn	
E CPU	0%	4 0x207f313:	pushl %edi	,op	
		5 0x207f314:	pushl %esi		
Memory	13.3 MB	6 0x207f315:	subl \$0x10, \$		
	100000000000	7 0x207f318: 8 0x207f31d:	calll 0x207f3	.0	; swift_dynamicCastObjCClassUnconditional + 13
		9 0x207f31e:	popl %esi movl 0x8(%ebp) Sedi	
Disk	Zero KB/s	10 0x207f321:	xorl %eax, %		
		11 0x207f323:	testl %edi, %		
Network	Zero KB/s	12 0x207f325:	je 0x207f34		; swift_dynamicCastObjCClassUnconditional + 54
•	1910 00000	13 0x207f327:	movl Øxc(%ebp		
_ 🚽 Thread 1		14 0x207f32a:		%esi), %ecx	
Queue: com.apple.mair	n-thread (serial)	15 0x207f330:	movl %eax, 0		
0 swift_dynamicCa	stObiCClassU	16 0x207f334:	movl %ecx, 0		
		17 0x207f338: 18 0x207f33b:	movl %edi, (% calll 0x208203		, sumbal stub for obje measand
I SMS_Demo.ContactData.getCo		18 0x2071330:	testb %al, %a		; symbol stub for: objc_msgSend
2 SMS_Demo.View	Controller.view	20 0x207f342:	movl %edi, %a		
			je 0x207f34		; swift_dynamicCastObjCClassUnconditional + 61
			addl \$0x10, \$		
*****	********************	23 0x207f349:	popl %esi	24	
23 UIApplicationMi	ain	24 0x207f34a:	popl %edi		
24 top_level_code		25 0x207f34b:	popl %ebp		
1 25 main		26 0x207f34c:	retl		
		A REAL PROPERTY AND A REAL			
	epatch_mana				
 25 main 26 start Thread 2 Queue: com.apple.libdi 	spatch-mana	27 0x207f34d: 28 0x207f353: 29 0x207f359: 30 0x207f35c:			