

# 使用 OllyDbg 从零开始 Cracking

## 第二章

(翻译: BGCode)

当我们回顾了 OllyDbg 的结构组成, 基本要素和原理后, 需要探究一下数制系统。

### 数制系统

最常用的数制系统是二进制, 十进制和十六进制。

了解它们, 最主要的你要知道是:

- 二进制: 只有符号 **0** 和 **1**, 因此称为二进制。
- 十进制: 出现 **10** 个字符 (从 **0** 到 **9**), 因此称为十进制。
- 十六进制: 从 **0** 到 **F** (**0-9**, **A**, **B**, **C**, **D**, **E** 和 **F**, 总共 **16** 个字符)。

通常, 除非另有说明, 当我们提及某一具体数字时都将其认作十六进制。

我们不使用将数值从一种数制转换为其它数制的令人不太愉快的数学公式。当前, Cracker 们一般使用 Windows 自带的计算器, 它将更加快捷和容易使用, 以避免繁冗的工作。

打开计算器, 准备工作。



进入菜单 View, 选择科学模式。



这里，我们看到默认的是十进制模式，在旁边还有其它三种进制可供选择，十六进制 **Hex**，八进制 **Oct**，二进制 **Bin**。

八进制使用 **8** 个字符，在 **Cracking** 中不太常用，但如果需要，也可在计算器中选择使用。

因此，将一个数字从一种数制系统转换到另一种数制系统，最简单的方法，先将计算器选择到数字初始数制的位置，例如，你需要将数字 **55** 从十进制转换为十六进制，在计算器位于十进制位置时输入 **55**。



现在将计算器换到十六进制符号的位置，结果将自动转换并显示出来。



这样，显然在十进制中的数字 **55** 转换为十六进制后为 **37**。

当使用在十进制中未出现的符号 **A**，**B**，**C**，**D**，**E**，**F** 时，我们可以从键盘输入这些信息。

我认为这种方法在实践中更有用途，允许我们不费力的将数字从一种数制转换到另一种数制。

## 十六进制负数

这有些难于理解，所以让我们从头道来。十六进制数制一定可以表示负数。如果不行话，那怎样表示一个相对应的十进制负数，例如 **-1**，用十六进制表示是什么？

考虑到这个问题后，我希望所有的问题将会逐渐变的明朗。

如果我们将 **00000000** 到 **FFFFFFFF** 所有可能的十六进制数都写出来，我们怎样表示负数？

我们将其中的一半表示正数，一半表示负数。

正数从 00000000 到 7FFFFFFF，负数从 80000000 到 FFFFFFFF

## 正数

00000000 和十进制 0 相同

00000001 仍然是十进制中的 1

.....

.....

7FFFFFFF 为十进制的 2147483647（也是最大正数）

## 负数

FFFFFFFF 和十进制-1 相同

FFFFFFFE 为十进制-2

.....

.....

80000000 等同于十进制的-2147483648（也是最小负数）

你可以试着用 CommandBar 插件查询 7FFFFFFF 在十进制中的值，只需在其前加问号，然后回车。



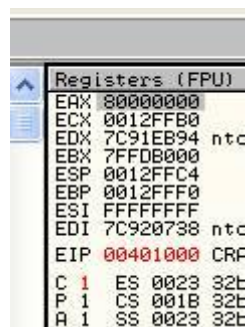
在右边，我们看到它返回了十进制值 2147483647，完全正确。

现在，我们想看看 80000000 的值是否为负，我们看到它不能显示（译注 1）7FFFFFFF 之后的值（这是 CommandBar 的一个 bug），那么在 OllyDbg 中如何检验它的值呢？

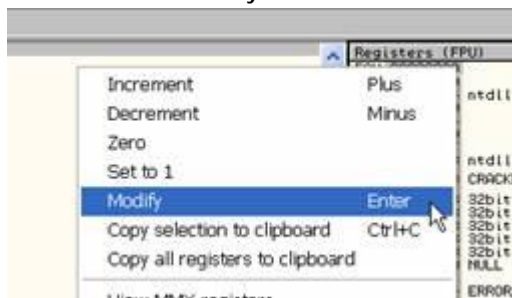


这是一个小技巧。

在寄存器窗口点选 EAX。

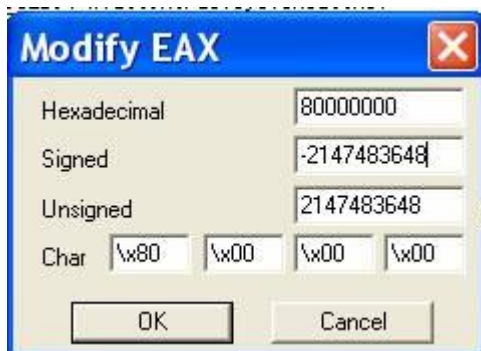


右键点击选择 **Modify**



出现的窗口让我们修改 **EAX** 的值，这个窗口也可以完成不同的转换功能。第一栏填入我们想转换的十六进制值，第二栏会出现相对应的十进制值。

这里，我们看到十六进制 **80000000** 转换为了十进制 **-2147483648**。



如果你想检验 **FFFFFFFF** 为十进制 **-1**。



在这个窗口中，可以更新寄存器。当我们轻松的验证完负数后，点击 **Cancel**。我们不要以任何方式改变寄存器的值。

## ASCII 字符

在以下截图中，看到的这种数据将是我们学习的内容之一。每个字符都被赋予了十六进制值。这允许我们将它们视为字符组合，字符和值等等。

这张表拷贝自(嘿嘿)«Теории ассемблера» (Caos Reptante)，你可以看到十进制值，相应的十

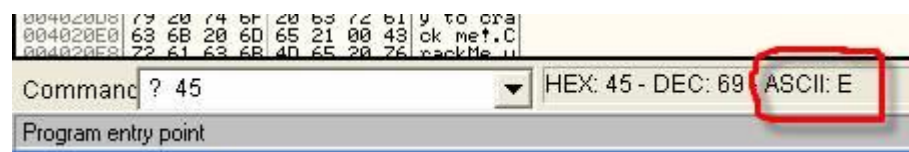
六进制值和字符。例如，你想在 OllyDbg 中使用空格，你可以使用 20 (Hex) 或者 32 (Dec)。

Dec.	Hex.	Carac	Dec.	Hex.	Carac	Dec.	Hex.	Caract
32	20	esp	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	\$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F		127	7F	□



另外，在 CommandBar 中，我们可以查询十六进制数字对应的字符。

? 45



我们看到 45 对应的是大写字母 E，如果你在上表中间一列查询 45，会发现它确实就是大写字母 E 的十六进制值。

69	45	E
----	----	---

在 OllyDbg 的数据 (Dump) 窗口中有一列为 ASCII 字符，让我们看看在 CrackMe (译注 2) 中是否出现了其中的一些字符。

Address	Hex dump	ASCII
00402000	00 00 00 00 00 00 00 00	.....
00402008	00 00 00 00 00 00 00 00	.....
00402010	00 00 00 00 00 00 00 00	.....
00402018	00 00 00 00 00 00 00 00	.....
00402020	00 00 00 00 00 00 00 00	.....
00402028	00 00 00 00 00 00 00 00	.....
00402030	00 00 00 00 00 00 00 00	.....
00402038	00 00 00 00 00 00 00 00	.....
00402040	00 00 00 00 00 00 00 00	.....
00402048	00 00 00 00 00 00 00 00	.....
00402050	00 00 00 00 00 00 00 00	.....
00402058	00 00 00 00 00 00 00 00	.....
00402060	00 00 00 00 00 00 00 00	.....
00402068	00 00 00 00 00 00 00 00	.....
00402070	00 00 00 00 00 00 00 00	.....
00402078	00 00 00 00 00 00 00 00	.....
00402080	00 00 00 00 00 00 00 00	.....
00402088	00 00 00 00 00 00 00 00	.....
00402090	00 00 00 00 00 00 00 00	.....
00402098	00 00 00 00 00 00 00 00	.....
004020A0	00 00 00 00 00 00 00 00	.....
004020A8	00 00 00 00 00 00 00 00	.....
004020B0	00 00 00 00 00 00 00 00	.....
004020B8	00 00 00 00 00 00 00 00	.....
004020C0	00 00 00 00 00 00 00 00	.....
004020C8	00 00 00 00 00 00 00 00	.....
004020D0	00 00 00 00 00 00 54 72	.....Tr
004020D8	79 20 74 6F 20 63 72 61	y to cra
004020E0	63 6B 20 6D 65 21 00 43	ck me!.C
004020E8	72 61 63 6B 4D 65 20 76	rackMe v
004020F0	31 2E 30 00 4E 6F 20 6E	1.0.No n
004020F8	65 65 64 20 74 6F 20 64	eed to d

在显示十六进制值那列的旁边，就是 ASCII 列，在那里，你可以看到由 ASCII 字符组成的文本串。

### 堆栈是什么？

它是内存的一块区域，用于短暂存储数据，这些数据稍后不久就要恢复取出。

就像在桌上放一叠信件或纸牌，最新的信件或纸牌都是放在最顶部，如果一张张地取走信件或纸牌，总会从最上面的开始取。

这是堆栈的主要性质，放在顶部的信件总会被最先取走。

以后我们将学习 OllyDbg 的堆栈怎样工作。

0012FFFC	7C81605F	RETURN to kernel32.7C81605F
0012FFFD	7C81605F	ntdll.7C807230
0012FFFE	7C81605F	.....
0012FFFF	7C81605F	.....
00130000	7C81605F	.....
00130001	7C81605F	.....
00130002	7C81605F	.....
00130003	7C81605F	.....
00130004	7C81605F	.....
00130005	7C81605F	.....
00130006	7C81605F	.....
00130007	7C81605F	.....
00130008	7C81605F	.....
00130009	7C81605F	.....
0013000A	7C81605F	.....
0013000B	7C81605F	.....
0013000C	7C81605F	.....
0013000D	7C81605F	.....
0013000E	7C81605F	.....
0013000F	7C81605F	.....
00130010	7C81605F	.....
00130011	7C81605F	.....
00130012	7C81605F	.....
00130013	7C81605F	.....
00130014	7C81605F	.....
00130015	7C81605F	.....
00130016	7C81605F	.....
00130017	7C81605F	.....
00130018	7C81605F	.....
00130019	7C81605F	.....
0013001A	7C81605F	.....
0013001B	7C81605F	.....
0013001C	7C81605F	.....
0013001D	7C81605F	.....
0013001E	7C81605F	.....
0013001F	7C81605F	.....
00130020	7C81605F	.....
00130021	7C81605F	.....
00130022	7C81605F	.....
00130023	7C81605F	.....
00130024	7C81605F	.....
00130025	7C81605F	.....
00130026	7C81605F	.....
00130027	7C81605F	.....
00130028	7C81605F	.....
00130029	7C81605F	.....
0013002A	7C81605F	.....
0013002B	7C81605F	.....
0013002C	7C81605F	.....
0013002D	7C81605F	.....
0013002E	7C81605F	.....
0013002F	7C81605F	.....
00130030	7C81605F	.....
00130031	7C81605F	.....
00130032	7C81605F	.....
00130033	7C81605F	.....
00130034	7C81605F	.....
00130035	7C81605F	.....
00130036	7C81605F	.....
00130037	7C81605F	.....
00130038	7C81605F	.....
00130039	7C81605F	.....
0013003A	7C81605F	.....
0013003B	7C81605F	.....
0013003C	7C81605F	.....
0013003D	7C81605F	.....
0013003E	7C81605F	.....
0013003F	7C81605F	.....
00130040	7C81605F	.....
00130041	7C81605F	.....
00130042	7C81605F	.....
00130043	7C81605F	.....
00130044	7C81605F	.....
00130045	7C81605F	.....
00130046	7C81605F	.....
00130047	7C81605F	.....
00130048	7C81605F	.....
00130049	7C81605F	.....
0013004A	7C81605F	.....
0013004B	7C81605F	.....
0013004C	7C81605F	.....
0013004D	7C81605F	.....
0013004E	7C81605F	.....
0013004F	7C81605F	.....
00130050	7C81605F	.....
00130051	7C81605F	.....
00130052	7C81605F	.....
00130053	7C81605F	.....
00130054	7C81605F	.....
00130055	7C81605F	.....
00130056	7C81605F	.....
00130057	7C81605F	.....
00130058	7C81605F	.....
00130059	7C81605F	.....
0013005A	7C81605F	.....
0013005B	7C81605F	.....
0013005C	7C81605F	.....
0013005D	7C81605F	.....
0013005E	7C81605F	.....
0013005F	7C81605F	.....
00130060	7C81605F	.....
00130061	7C81605F	.....
00130062	7C81605F	.....
00130063	7C81605F	.....
00130064	7C81605F	.....
00130065	7C81605F	.....
00130066	7C81605F	.....
00130067	7C81605F	.....
00130068	7C81605F	.....
00130069	7C81605F	.....
0013006A	7C81605F	.....
0013006B	7C81605F	.....
0013006C	7C81605F	.....
0013006D	7C81605F	.....
0013006E	7C81605F	.....
0013006F	7C81605F	.....
00130070	7C81605F	.....
00130071	7C81605F	.....
00130072	7C81605F	.....
00130073	7C81605F	.....
00130074	7C81605F	.....
00130075	7C81605F	.....
00130076	7C81605F	.....
00130077	7C81605F	.....
00130078	7C81605F	.....
00130079	7C81605F	.....
0013007A	7C81605F	.....
0013007B	7C81605F	.....
0013007C	7C81605F	.....
0013007D	7C81605F	.....
0013007E	7C81605F	.....
0013007F	7C81605F	.....
00130080	7C81605F	.....
00130081	7C81605F	.....
00130082	7C81605F	.....
00130083	7C81605F	.....
00130084	7C81605F	.....
00130085	7C81605F	.....
00130086	7C81605F	.....
00130087	7C81605F	.....
00130088	7C81605F	.....
00130089	7C81605F	.....
0013008A	7C81605F	.....
0013008B	7C81605F	.....
0013008C	7C81605F	.....
0013008D	7C81605F	.....
0013008E	7C81605F	.....
0013008F	7C81605F	.....
00130090	7C81605F	.....
00130091	7C81605F	.....
00130092	7C81605F	.....
00130093	7C81605F	.....
00130094	7C81605F	.....
00130095	7C81605F	.....
00130096	7C81605F	.....
00130097	7C81605F	.....
00130098	7C81605F	.....
00130099	7C81605F	.....
0013009A	7C81605F	.....
0013009B	7C81605F	.....
0013009C	7C81605F	.....
0013009D	7C81605F	.....
0013009E	7C81605F	.....
0013009F	7C81605F	.....
001300A0	7C81605F	.....
001300A1	7C81605F	.....
001300A2	7C81605F	.....
001300A3	7C81605F	.....
001300A4	7C81605F	.....
001300A5	7C81605F	.....
001300A6	7C81605F	.....
001300A7	7C81605F	.....
001300A8	7C81605F	.....
001300A9	7C81605F	.....
001300AA	7C81605F	.....
001300AB	7C81605F	.....
001300AC	7C81605F	.....
001300AD	7C81605F	.....
001300AE	7C81605F	.....
001300AF	7C81605F	.....
001300B0	7C81605F	.....
001300B1	7C81605F	.....
001300B2	7C81605F	.....
001300B3	7C81605F	.....
001300B4	7C81605F	.....
001300B5	7C81605F	.....
001300B6	7C81605F	.....
001300B7	7C81605F	.....
001300B8	7C81605F	.....
001300B9	7C81605F	.....
001300BA	7C81605F	.....
001300BB	7C81605F	.....
001300BC	7C81605F	.....
001300BD	7C81605F	.....
001300BE	7C81605F	.....
001300BF	7C81605F	.....
001300C0	7C81605F	.....
001300C1	7C81605F	.....
001300C2	7C81605F	.....
001300C3	7C81605F	.....
001300C4	7C81605F	.....
001300C5	7C81605F	.....
001300C6	7C81605F	.....
001300C7	7C81605F	.....
001300C8	7C81605F	.....
001300C9	7C81605F	.....
001300CA	7C81605F	.....
001300CB	7C81605F	.....
001300CC	7C81605F	.....
001300CD	7C81605F	.....
001300CE	7C81605F	.....
001300CF	7C81605F	.....
001300D0	7C81605F	.....
001300D1	7C81605F	.....
001300D2	7C81605F	.....
001300D3	7C81605F	.....
001300D4	7C81605F	.....
001300D5	7C81605F	.....
001300D6	7C81605F	.....
001300D7	7C81605F	.....
001300D8	7C81605F	.....
001300D9	7C81605F	.....
001300DA	7C81605F	.....
001300DB	7C81605F	.....
001300DC	7C81605F	.....
001300DD	7C81605F	.....
001300DE	7C81605F	.....
001300DF	7C81605F	.....
001300E0	7C81605F	.....
001300E1	7C81605F	.....
001300E2	7C81605F	.....
001300E3	7C81605F	.....
001300E4	7C81605F	.....
001300E5	7C81605F	.....
001300E6	7C81605F	.....
001300E7	7C81605F	.....
001300E8	7C81605F	.....
001300E9	7C81605F	.....
001300EA	7C81605F	.....
001300EB	7C81605F	.....
001300EC	7C81605F	.....
001300ED	7C81605F	.....
001300EE	7C81605F	.....
001300EF	7C81605F	.....
001300F0	7C81605F	.....
001300F1	7C81605F	.....
001300F2	7C81605F	.....
001300F3	7C81605F	.....
001300F4	7C81605F	.....
001300F5	7C81605F	.....
001300F6	7C81605F	.....
001300F7	7C81605F	.....
001300F8	7C81605F	.....
001300F9	7C81605F	.....
001300FA	7C81605F	.....
001300FB	7C81605F	.....
001300FC	7C81605F	.....
001300FD	7C81605F	.....
001300FE	7C81605F	.....
001300FF	7C81605F	.....



好的，我认为这次的课程应该结束了。在第三章，我们将学习寄存器，标志以及它们的意义。

#### 译注 1

CommandBar 插件只能显示到 7FFFFFFF 的值，输入 7FFFFFFF 之后的任意数值，它也只能显示 7FFFFFFF 的结果。

#### 译注 2

本文使用第一章的 CrackMe，包含在随文附件中

随文附件

1. CrackMe: ollydbg01-Crackme.zip

#### 翻译说明：

该系列教程目前官方已更新到第 47 章。本文原文为俄语，译者不才，斗胆翻译，采用了能用的所有手段。虽经本人严加审校，但难免讹误。有些词句加入了译者的理解，所以部分内容可能与原文有所出入。翻译本文也是译者学习的过程，所以错误在所难免。如发现错误，敬请指正，以免误人子弟。

该系列教程链接：<http://wasm.ru/series.php?sid=17>

本文原文链接：<http://wasm.ru/article.php?article=ollydbg02>

本文原文版权：[C] Рикардо Нарваха, пер. Aquila

译文版权：BGCoder, <http://www.pediy.com/>