CØDE

HACK THE HOLIDAYS 2018

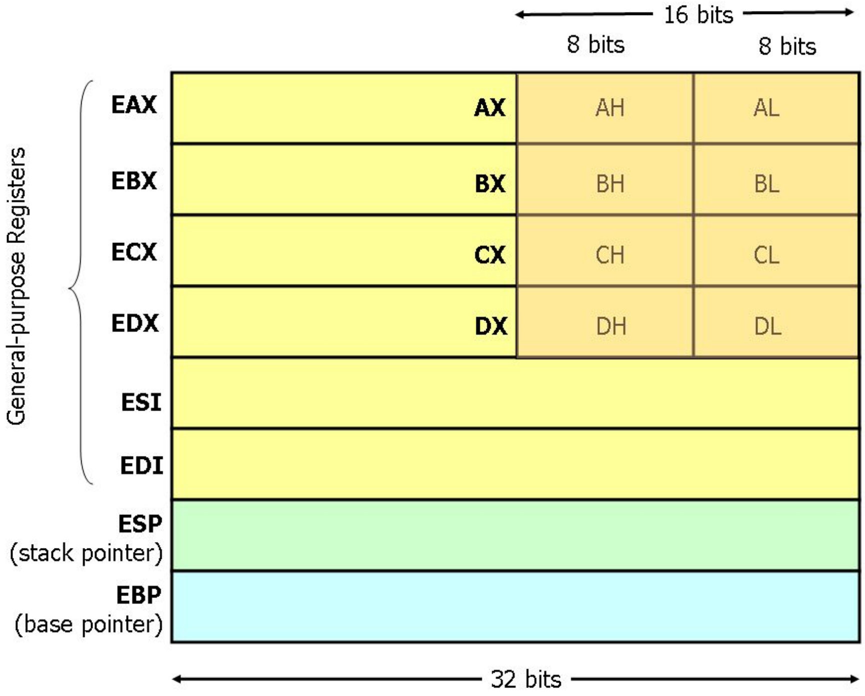
Workshop Handout

12/10/2018

x86 Basics

There is an insane number of registers in the x86 architecture when you consider all special purpose registers. Some are so obscure you can RE for years and never encounter or use them. Thankfully, there are just 8 you need to know to get started.

General Purpose Registers



Back in 16-bit land, we had registers like AX, BX, CX, etc. When 32-bit came around, they were ‘extended’ to EAX, EBX, ECX, etc. This is where the E comes from.

EAX, EBX, ECX, and EDX are “special” in that you can address the lower two bytes directly - you cannot do this with other registers.

Example: If EAX is 0x11223344, the AX is 0x3344, AH is 0x33, and AL is 0x44.

Intel Syntax

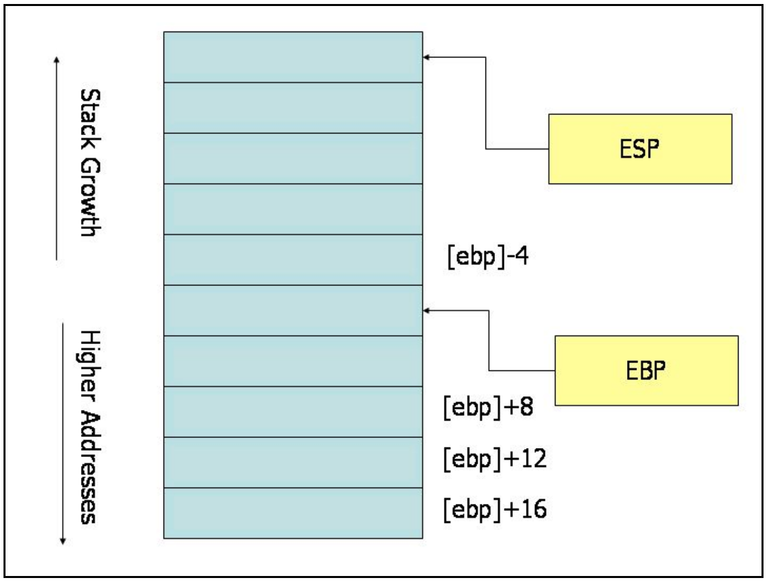
There are two main syntactic flavors of x86; Intel and AT&T. In most of today's applications and documentation, Intel is favored. Everything you'll see in this workshop is Intel. One important thing to know about Intel syntax is that it is *destination first*.

```
ins dest, src

mov eax, 0x01   (eax = 1 after execution)

add ebx, ecx    (ebx = ebx + ecx after execution)
```

The Stack

The stack is an area of memory used within function scope - meaning each function has its own "stack frame" that's created and collapsed as execution enters and exits a function. Here are some interesting this to consider:

- The stack "grows towards lower addresses"
- Subtracting from the stack pointer "creates" space in the frame
- Adding to the stack pointer "removes" space from the frame
- Values on the stack stay there, even when a frame is collapsed



Bits, Bytes & Endianness

Data can be represented in a variety of ways. Data types, lengths, encodings all have unique names people use to refer to them. Here we cover a few of the most common.

Bits and Bytes

```
4 bits  = 1 nibble (0000 0000, think ax, bx, cx, etc.)
8 bits  = 1 byte
16 bits = 2 bytes = 1 WORD
32 bits = 4 bytes = 2 WORDS = 1 DWORD (double word)
64 bits = 8 bytes = 4 WORDS = 2 DWORD = 1 QWORD (quad word)
```

32-bit systems have a 32-bit address space. The max value that can be represented is 4,294,967,295 (4 billion).

64-bit systems have a 64-bit address space. The max value that can be represented is 18,446,744,073,709,551,615 (18 quintillion).

Endianness

x86 is a little-endian architecture, unfortunately, humans think in big-endian. So what is endianness? It's the method used when ordering sequential bytes.

There are two ways multi-byte values are stored and referenced, big-endian (most intuitive for humans) and little-endian. Little endian "swaps" byte ordering so the *least* significant byte comes first.

Example: Let's say we want to store 44,596 as a WORD (16-bits, 2-bytes). In big-endian, this is 0xAE 0x34. But in little-endian architectures (like x86), the value 44,596 is stored as 0x34 0xAE.

Example: If we expand out to a DWORD for a larger value like 1,352,398, big-endian would be 0x00 0x14 0xA2 0xCE, while little endian would be 0xCE 0xA2 0x14 0x00. Crazy huh?

Some architectures like MIPS, have big and little-endian flavors. They are generally referred to as MIPSEB (endian, big) and MIPSEL (endian, little).

Binary Ninja

For this workshop, we'll be using a disassembler called Binary Ninja. A disassembler reversed the assembly operations performed by compilers to bring machine code back into human readable instruction set architecture mnemonics. Many of today's disassemblers offer much more functionality than mere disassembly, including decompilation, intermediate language lifting, plugin support, development APIs, etc.

Useful Hotkeys

Hot Key	Description
ctrl + [	Navigate backward
ctrl +]	Navigate forward
esc	Navigate backward
g	Goto an address
h	Switch to hex view
i	Cycle between Assembly, LLIL, MLIL
n	Rename the selected element (function, variable, etc.)
r	Change selected byte to an ASCII character
space	Switch between linear and graph modes (IL only avail. In graph)
;	Add a comment

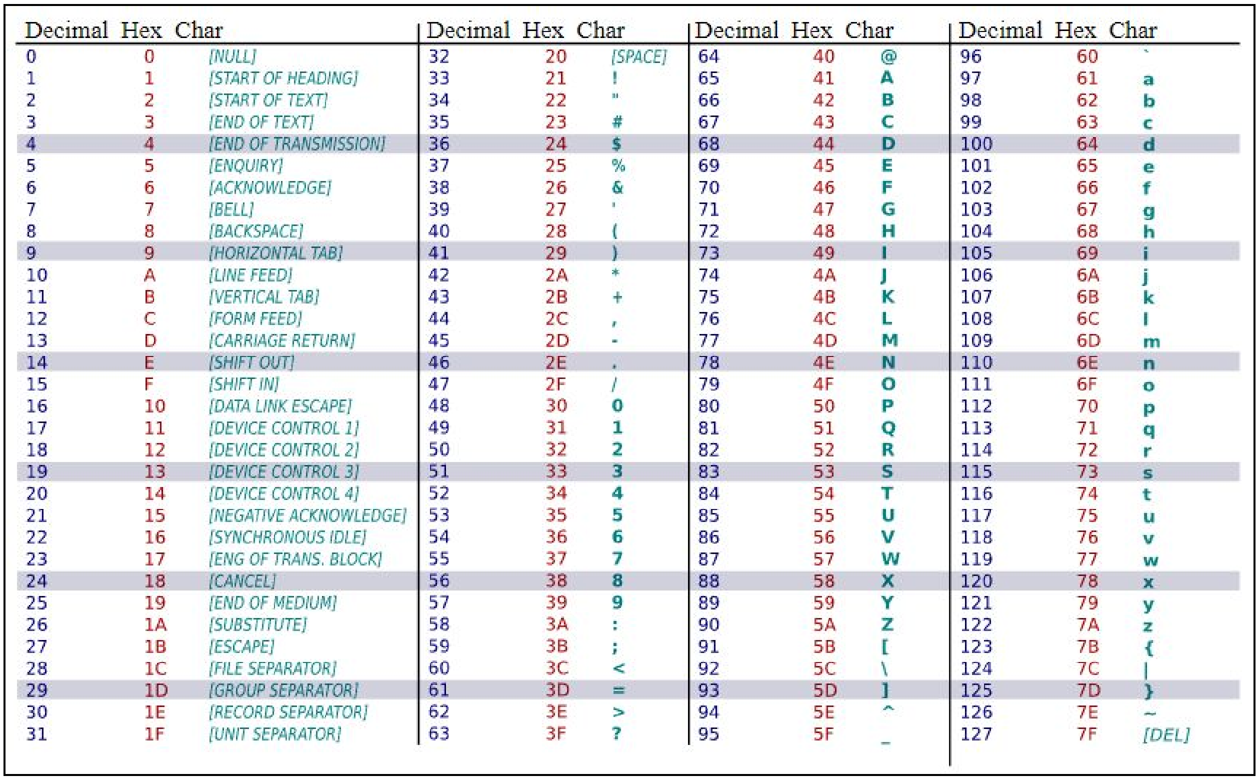
x64dbg

x64dbg is one of many debuggers available to debug Windows Portable Executable (PE) files. x64dbg is the spiritual successor to the extremely popular OllyDBG and Immunity debuggers with a ton of useful additions. x64dbg is quite a bit more full-featured than typical debuggers - including a decompiler (snowman), graph view, and YARA integration.

Useful Hotkeys

Hot Key	**Description**
F2	Set / Unset Breakpoint
F7	Single step into
F8	Single step over
F9	Run
ctrl-F9	Run until return (useful when you accidentally step into)
ctrl -g	Goto an address
space	Assemble an instruction (choose ‘fill with nops’)

ASCII Table



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]