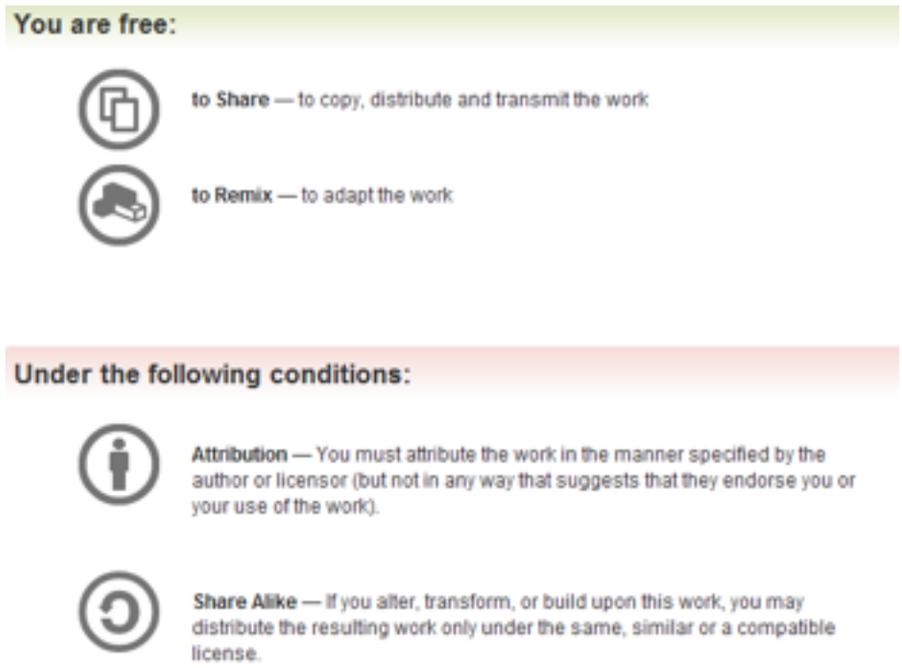
Introduction to Intel x86-64 Assembly, Architecture, Applications, & Alliteration

Xeno Kovah – 2014-2015
xeno@legbacore.com

All materials is licensed under a Creative Commons “Share Alike” license.

- http://creativecommons.org/licenses/by-sa/3.0/



Attribution condition: You must indicate that derivative work
"Is derived from Xeno Kovah's 'Intro x86-64' class, available at http://OpenSecurityTraining.info/IntroX86-64.html"

Attribution condition: You must indicate that derivative work

"Is derived from Xeno Kovah's ‘Intro x86-64’ class, available at http://OpenSecurityTraining.info/IntroX86-64.html"

Refresher - Data Types

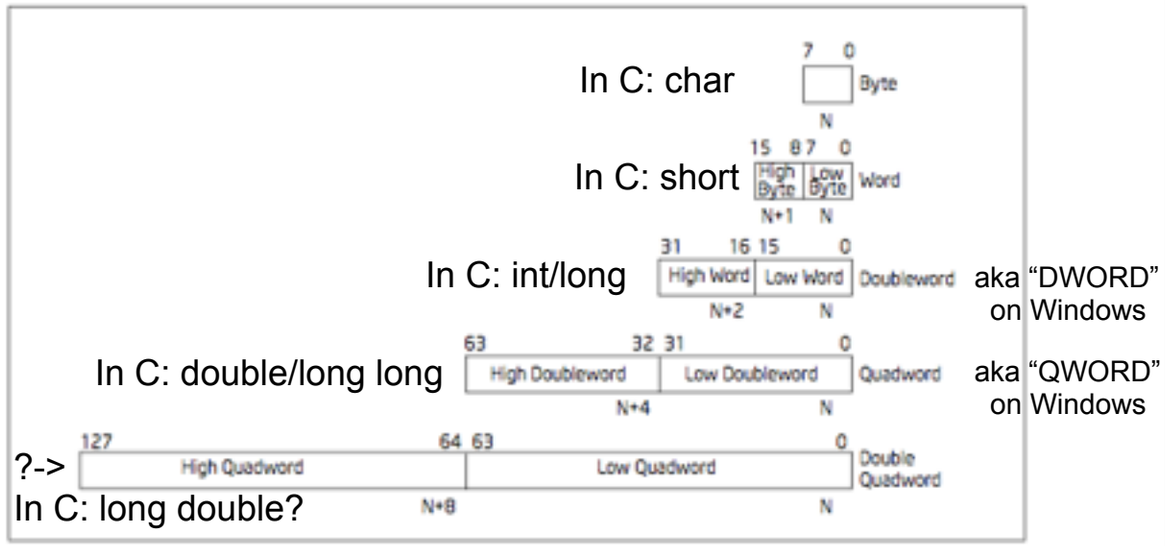


Figure 4-1. Fundamental Data Types

Intel Vol 1 Sec 4.1 - page 4-1 -

(All citations will be from the included June 2014 manual, because things move around slightly between revisions)

Refresher - Alt. Radices
Decimal, Binary, Hexidecimal

If you don’t know this, you must memorize tonight

Decimal (base 10)	Binary (base 2)	Hex (base 16)
00	0000b	0x00
01	0001b	0x01
02	0010b	0x02
03	0011b	0x03
04	0100b	0x04
05	0101b	0x05
06	0110b	0x06
07	0111b	0x07
08	1000b	0x08
09	1001b	0x09
10	1010b	0x0A
11	1011b	0x0B
12	1100b	0x0C
13	1101b	0x0D
14	1110b	0x0E
15	1111b	0x0F

Maybe go practice here?
http://forums.cisco.com/CertCom/game/binary_game_page.htm

Refresher - Negative Numbers

- Negative numbers are defined as the “two’s complement” of the positive number
- “one’s complement” = flip all bits. 0->1, 1->0
- “two’s complement” = one’s complement + 1

Number	One’s Comp.	Two’s Comp. (negative)
00000001b : 0x01	11111110b : 0xFE	11111111b : 0xFF : -1
00000100b : 0x04	11111011b : 0xFB	11111100b : 0xFC : -4
00011010b : 0x1A	11100101b : 0xE5	11100110b : 0xE6 : -26
?	?	10110000b : 0xB0 : -?

- 0x01 to 0x7F positive byte, 0x80 to 0xFF negative byte
- 0x00000001 to 0x7FFFFFFF positive dword
- 0x80000000 to 0xFFFFFFFF negative dword
- 0x0000000000000001 to 0x7FFFFFFFFFFFFFFF positive dword
- 0x8000000000000000 to 0xFFFFFFFFFFFFFFFF negative dword

Refresher - Boolean (“bitwise”) logic

AND “&”

0	0	0
0	1	0
1	0	0
1	1	1

Operands **Result**

OR “|”

0	0	0
0	1	1
1	0	1
1	1	1

XOR “^”

0	0	0
0	1	1
1	0	1
1	1	0

NOT “~”

0	1
1	0