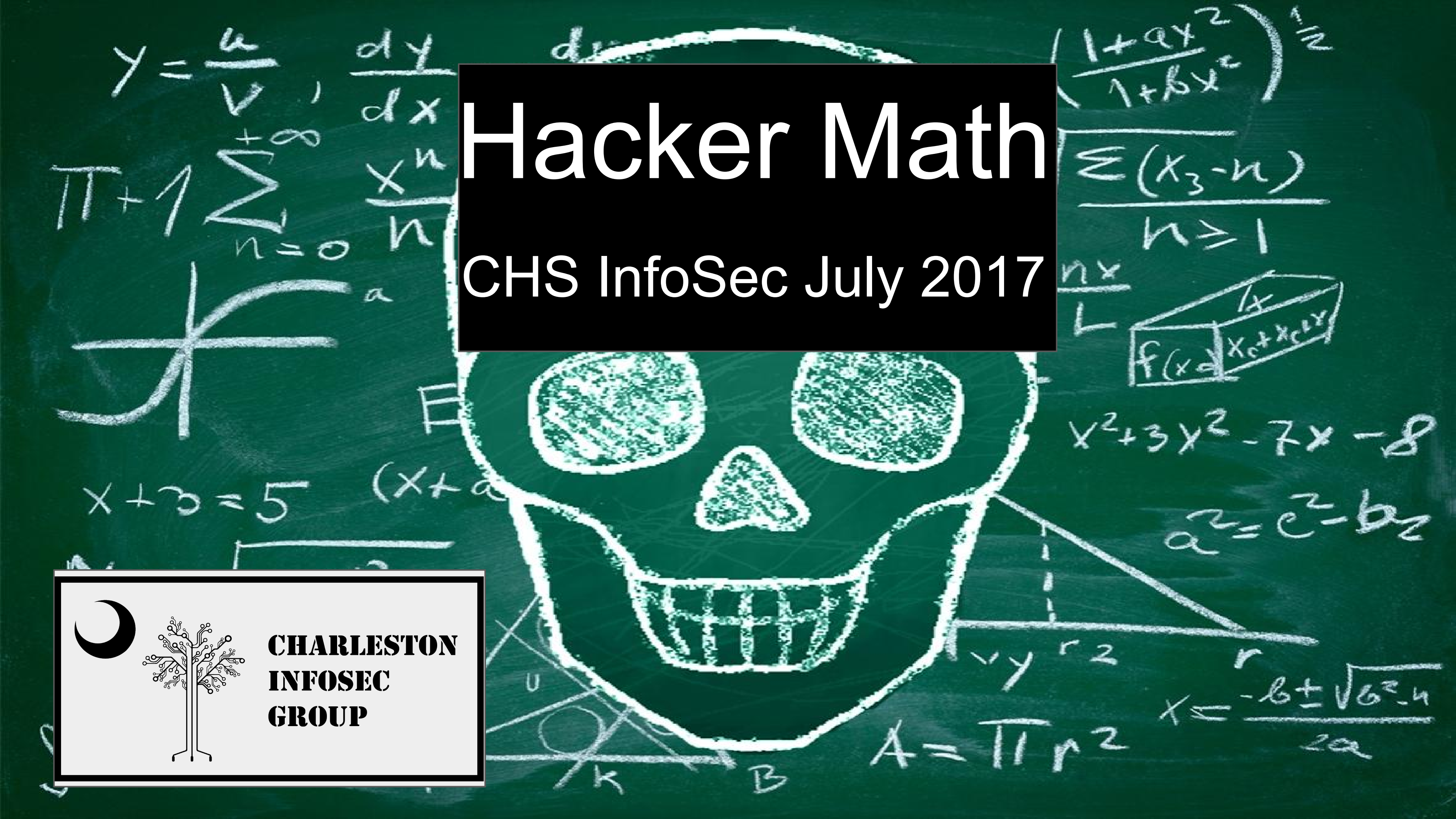


Hacker Math

CHS InfoSec July 2017

CHARLESTON
INFOSEC
GROUP

Numbers are important!

Numbers are important in our world in fact number systems are the base of reason for a lot of things around us.

What is your age?

When is your birthday?

What's your phone number?

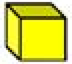
All of the answers to those questions are related to numbers..

Number Systems and Base-10 (Decimal)

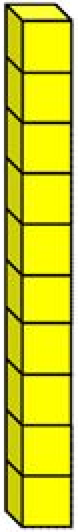
Number systems are the way we count and represent numbers. There are many different number systems that use different bases (digit size). The one we are most used to is Base-10 (**decimal**).

In decimal, each digit in a position of a number can hold 10 numbers ranging from 0 to 9 (10 possibilities). The places or positions of the numbers are based on powers of ten (e.g., hundredths, tenths, tens, hundreds, thousands). Exceeding the number 9 in a position starts counting in the next highest position.

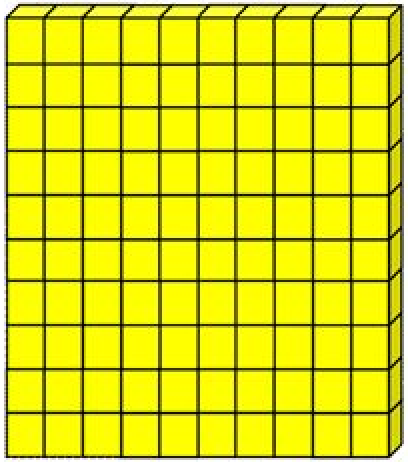
Can you figure why where Base-10 came from?
(raise your hand)



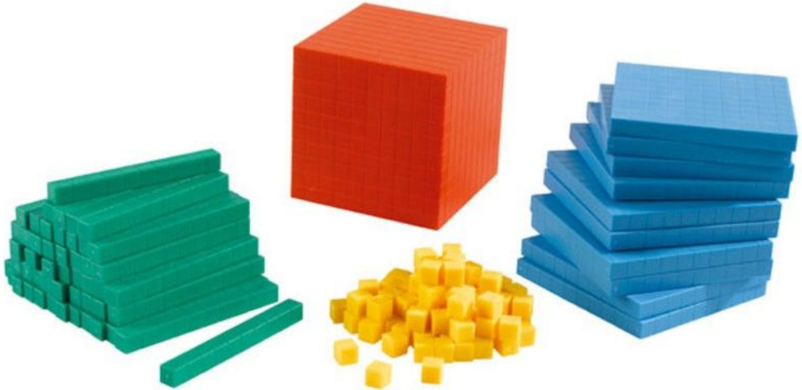
= 1



=10



=100

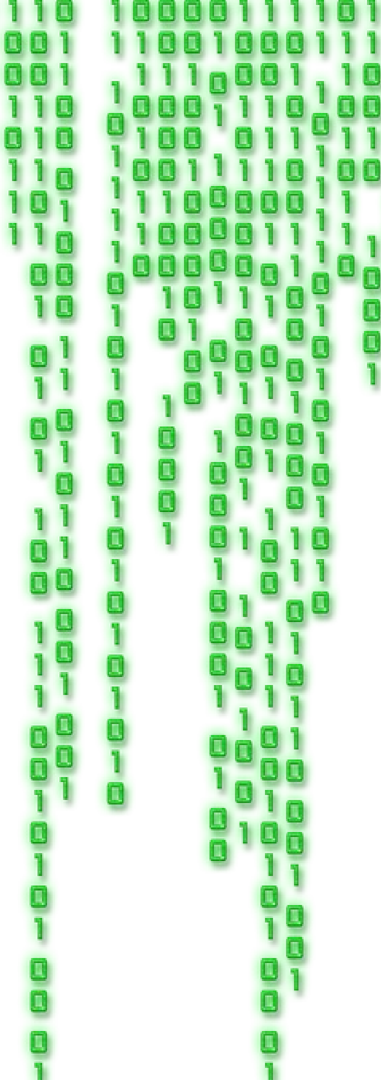


Base-2 (Binary)

Now computers at its lowest level uses a different number system called Base-2 also known as **binary**. This is the stuff you see “in the matrix”

In binary, each digit can hold 2 numbers so 0-1. This means counting in binary should look like this:

Decimal	0	1	2	3	4	5
Binary	0	1	10	11	100	101



Place values

(multiply this number by the 1 or 0 in its place)

128	64	32	16	8	4	2	1
×	×	×	×	×	×	×	×
1	0	1	1	0	1	0	1
=	=	=	=	=	=	=	=

128 + 0 + 32 + 16 + 0 + 4 + 0 + 1

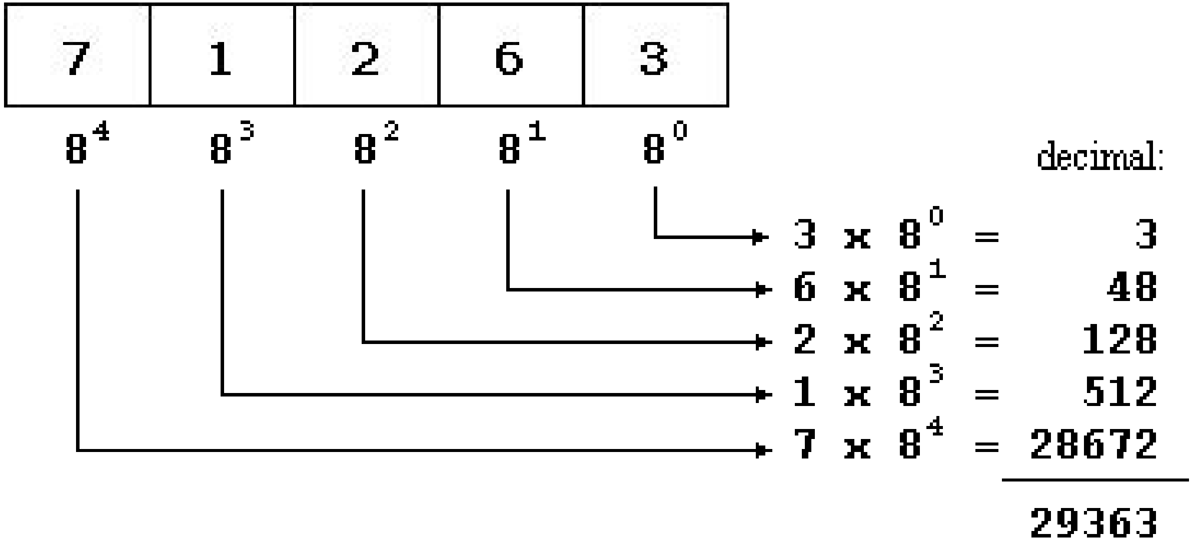
(add all these together to get the decimal number)

=

181

Base-8 (Octal)

Base-8 or **octal** uses digits that can old 8 numbers 0-7 (get the pattern here?). We will not go into this one but it is mostly used in early computing and aviation.



7	1	2	6	3
8^4	8^3	8^2	8^1	8^0

decimal:

$3 \times 8^0 = 3$

$6 \times 8^1 = 48$

$2 \times 8^2 = 128$

$1 \times 8^3 = 512$

$7 \times 8^4 = 28672$

29363

Base-16 (Hexadecimal)

Base-16 or **hexadecimal** numerals are widely used by computer system designers and programmers. It has a base digit that holds 16 numbers that is 0-15.

Now you be be noticing a problem here.

10-15 are 2 digits!... how will this ever make sense!

Mathmaticians came up with another way to represent numbers 10-15 using the alphabet.

More HEX

10 = A

11 = B

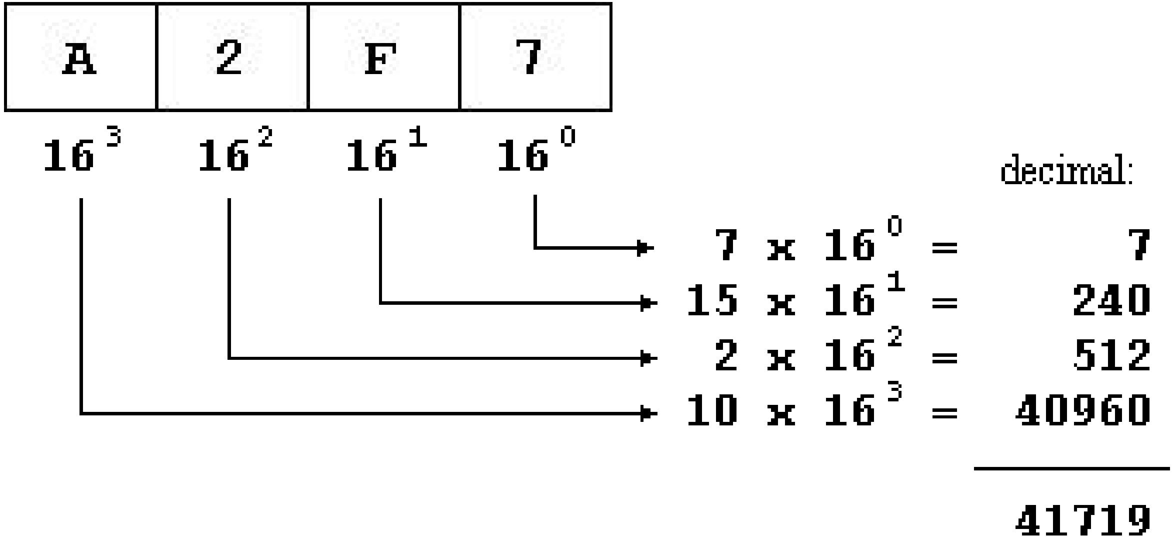
12 = C

13 = D

14 = E

15 = F

This allows for a single digit to represent 0-15 respectively.



$7 \times 16^0 = 7$

$15 \times 16^1 = 240$

$2 \times 16^2 = 512$

$10 \times 16^3 = 40960$

41719

python
powered

```
print("Hello, world!")
```

Doing conversions in python and bitwise operations

Go to http://bit.ly/chsinfosec-hackermath to follow along!

» DATA «
WWW.

Units of data measurement

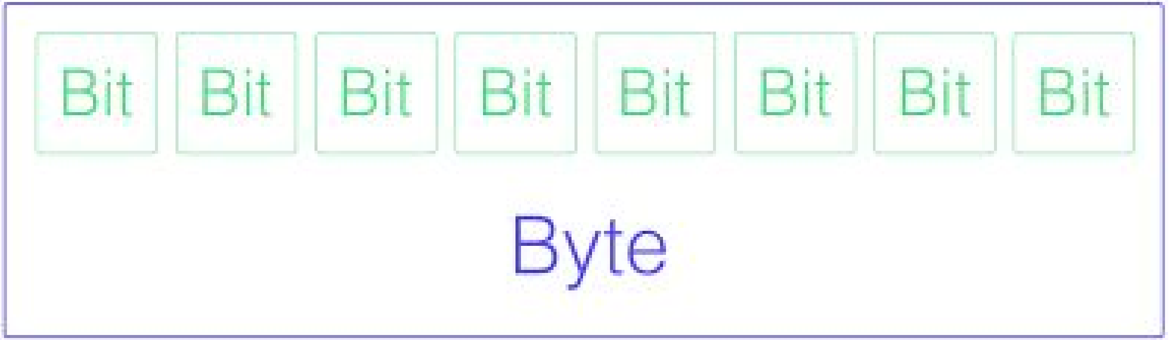
In computing and telecommunications, a unit of information is the capacity of some standard data storage system or communication channel, used to measure the capacities of other systems and channels. In information theory, units of information are also used to measure the entropy of random variables and information contained in messages.

The most commonly used units of data storage capacity are the bit, the capacity of a system that has only two states, and the byte (or octet), which is equivalent to eight bits.

Bytes (8 Bits)

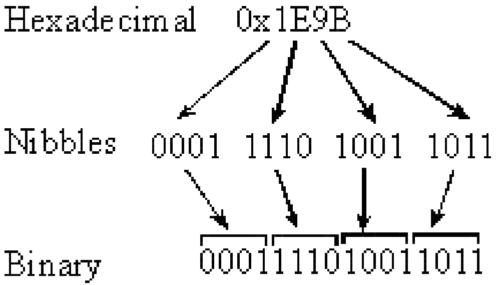
Historically, a byte was the number of bits used to encode a character of text in the computer, which depended on computer hardware architecture; but today it almost always means **8 bits** – that is, an octet.

A byte can represent 256 (28) distinct values, such as the integers 0 to 255, or −128 to 127.



Nibble… YUM! (4 Bits)

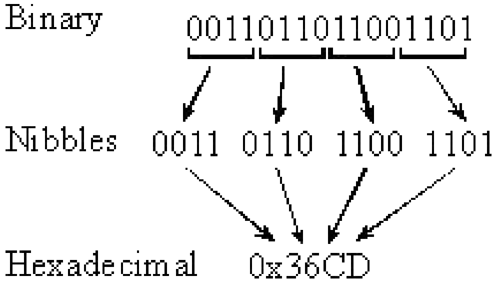
A group of **4 bits**, or half a byte, is sometimes called a nibble or nybble. This unit is most often used in the context of hexadecimal number representations, since a nibble has the same amount of information as one hexadecimal digit.



Hexadecimal 0x1E9B

Nibbles 0001 1110 1001 1011

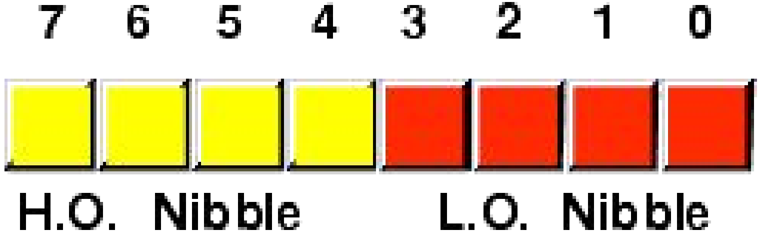
Binary 0001111010011011



Binary 0011011011001101

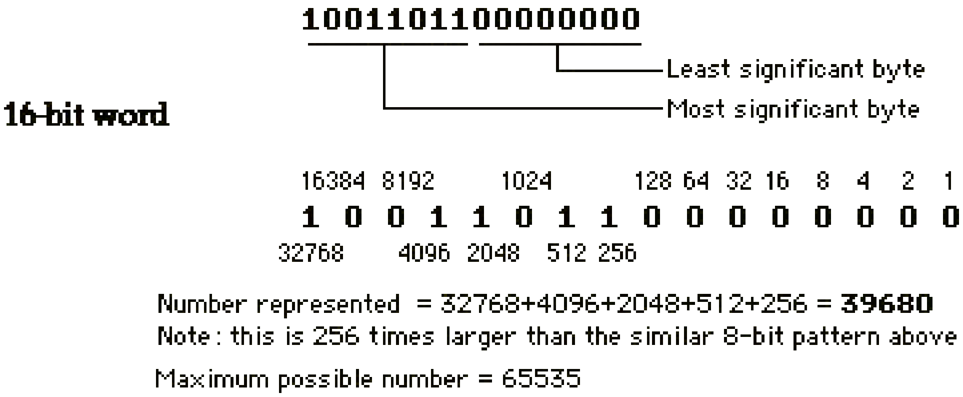
Nibbles 0011 0110 1100 1101

Hexadecimal 0x36CD



Word (16 bits)

Computers usually manipulate bits in groups of a fixed size, conventionally called words. The number of bits in a word is usually defined by the size of the registers in the computer's CPU, or by the number of data bits that are fetched from its main memory in a single operation. In the IA-32 architecture more commonly known as x86-32, a word is **16 bits**.

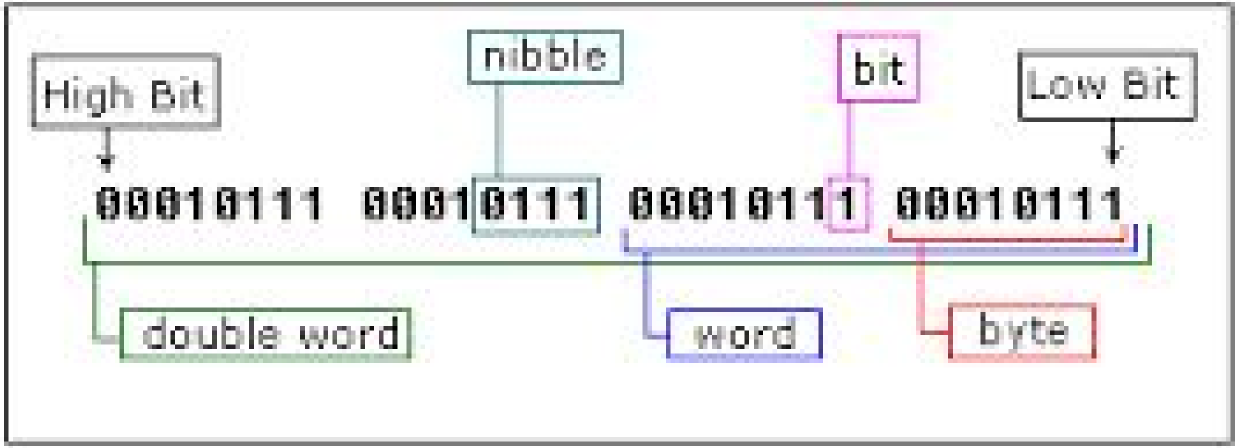


Number represented = 32768+4096+2048+512+256 = **39680**
Note: this is 256 times larger than the similar 8-bit pattern above

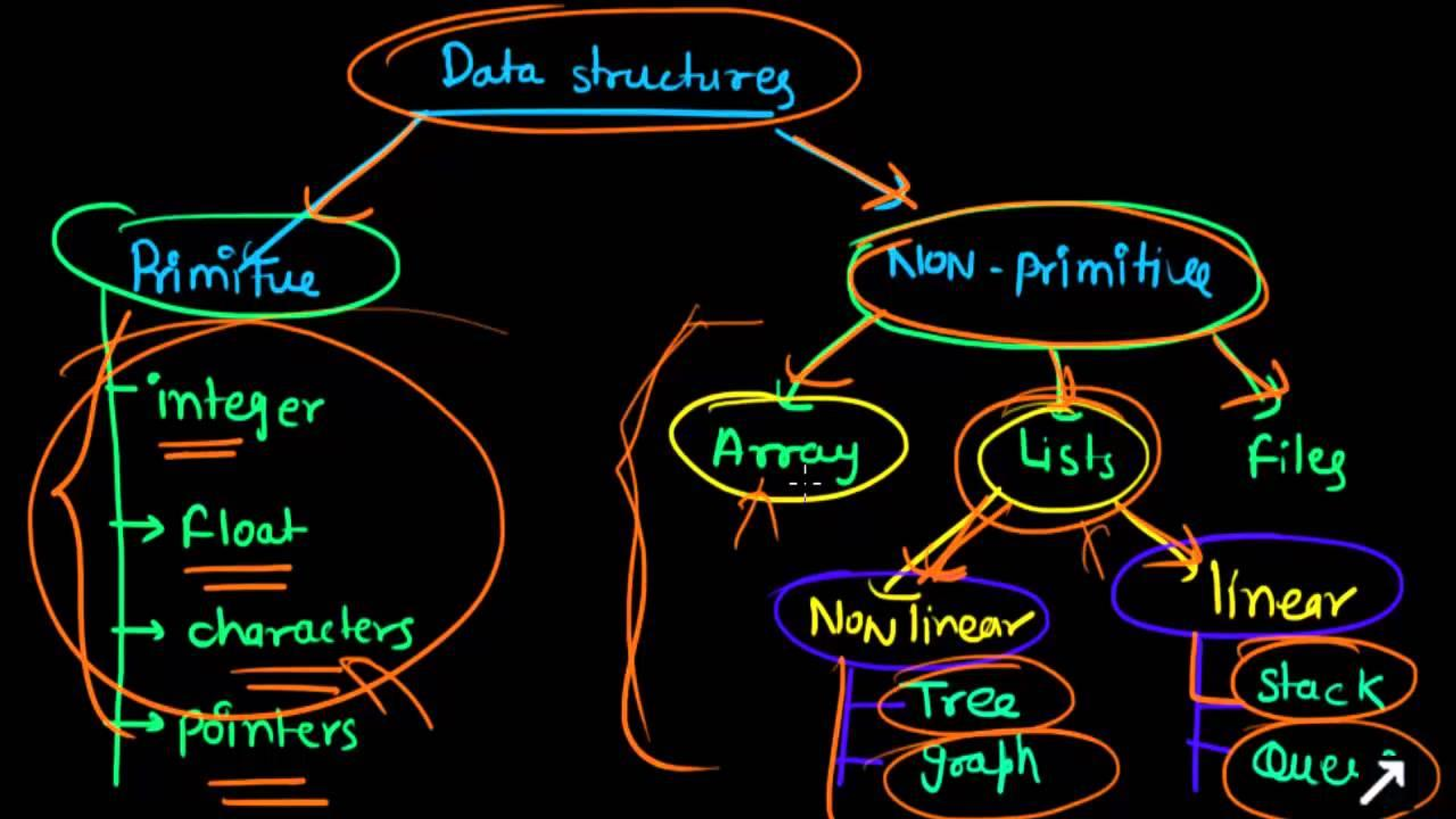
Maximum possible number = 65535

DWord (32 bits)

DWord stands for double word which as you guess is the same as a word but 32 bits instead on x86-32.



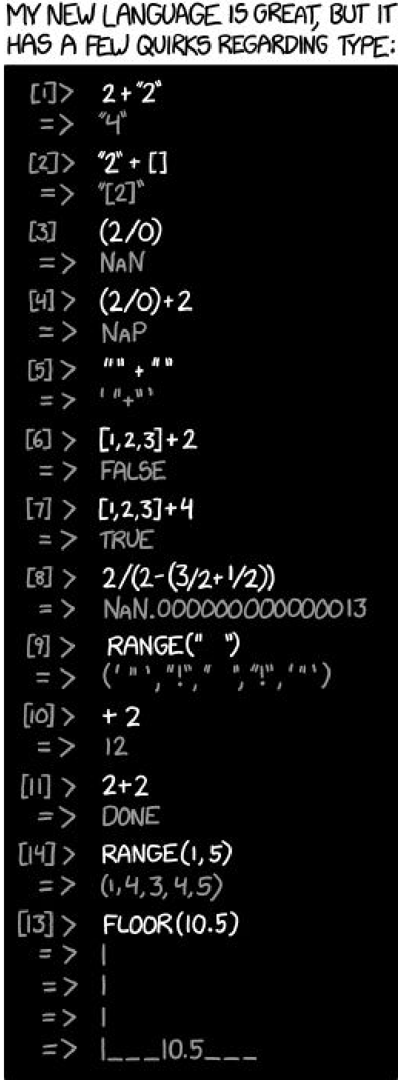
	Size in Bits	Size in Bytes
Bits	1	
Nibble	4	
Byte	8	1
Word	16	2
DWord	32	4


Data structures
Primitive
NON - primitive
integer
float
characters
pointers
Array
Lists
Files
Non linear
linear
Tree
graph
Stack
Queue

Primitive Data Structure Types

Data types are used within type systems, which offer various ways of defining, implementing and using them. Different type systems ensure varying degrees of type safety.

A computer’s way of saying this is what I expect this to be


MY NEW LANGUAGE IS GREAT, BUT IT
HAS A FEW QUIRKS REGARDING TYPE:

```
[1]>  2+"2"
 =>   "4"
[2]>  "2"+[]
 =>   "[2]"
[3]   (2/0)
 =>   NaN
[4]>  (2/0)+2
 =>   NaP
[5]>  "" + ""
 =>   '"+"'
[6]>  [1,2,3]+2
 =>   FALSE
[7]>  [1,2,3]+4
 =>   TRUE
[8]>  2/(2-(3/2+1/2))
 =>   NaN.0000000000000013
[9]>  RANGE(" ")
 =>   ('"',"!"," ","!",'"')
[10]> +2
 =>   12
[11]> 2+2
 =>   DONE
[14]> RANGE(1,5)
 =>   (1,4,3,4,5)
[13]> FLOOR(10.5)
 =>   |
 =>   |
 =>   |
 =>   |___10.5___
```

Integers

Represents a finite subset of mathematical numbers (0, 1, 2, …..)

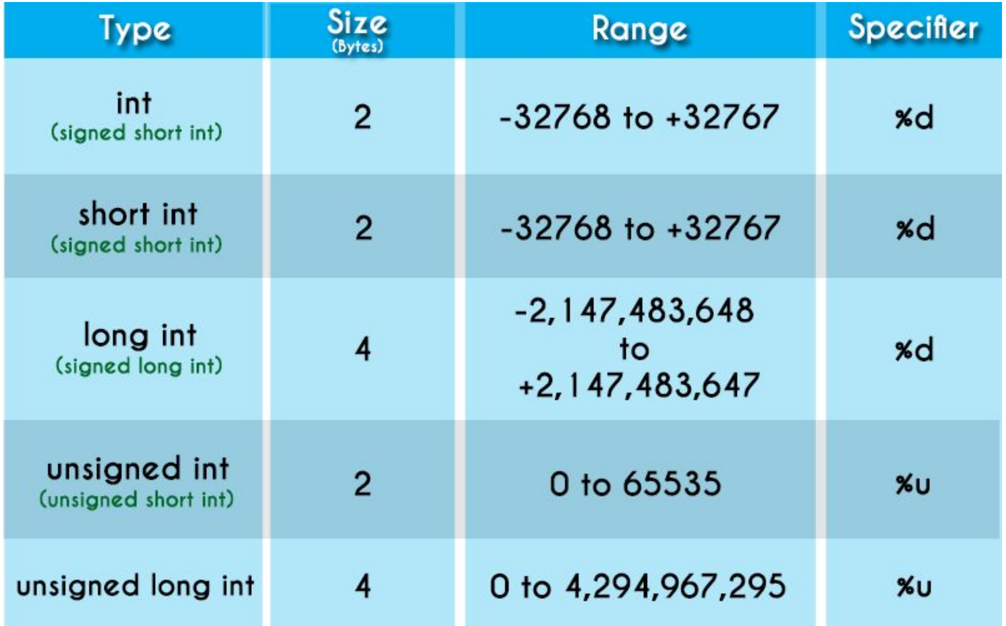
Comes in many flavors

- short / long
- signed/unsigned

Example:

(hex) FF would represent 255

(binary) 11111111 would represent 255



Type	Size (Bytes)	Range	Specifier
int (signed short int)	2	-32768 to +32767	%d
short int (signed short int)	2	-32768 to +32767	%d
long int (signed long int)	4	-2,147,483,648 to +2,147,483,647	%d
unsigned int (unsigned short int)	2	0 to 65535	%u
unsigned long int	4	0 to 4,294,967,295	%u

Boolean

This is a data type that denotes: (True/False), (On/Off) , (0,1)

1 = True and 0 = False

The Boolean data type is primarily associated with conditional statements, which allow different actions and change control flow depending on whether a programmer-specified Boolean condition evaluates to true or false.

Characters

a character is a unit of information that roughly corresponds to a character in the alphabet or symbol in a text language.

Characters can be represented in many ways but the usual methods are Unicode (UTF-8, UTF-16) and ASCII.

All of which use different sizes and map numbers to different characters.

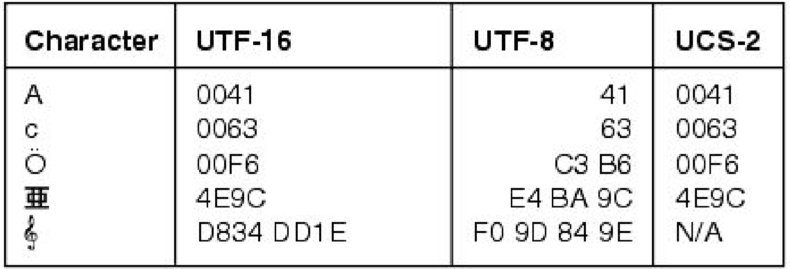
Unicode (UTF-8 and UTF-16)

Shifting from UTF-8 -> UTF-16

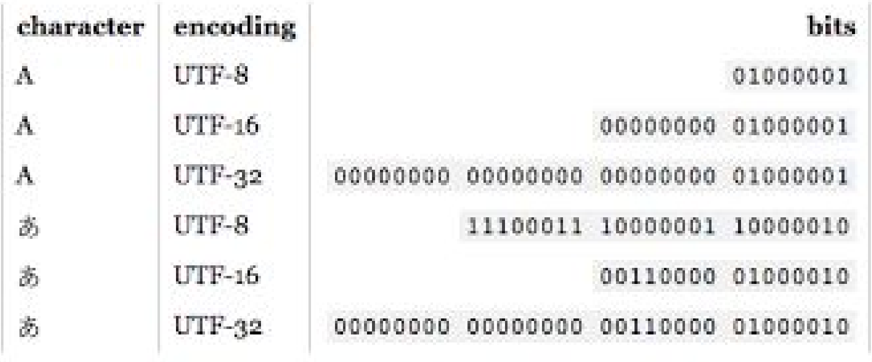
UTF-8 = 1 byte while UTF-16 = 2 Bytes

As you can see 01000001 (41) is a capital A in UTF-8 and in UTF-16

The reason for more bits is to handle more characters without having to add more encodings.



Character	UTF-16	UTF-8	UCS-2
A	0041	41	0041
c	0063	63	0063
Ö	00F6	C3 B6	00F6
亜	4E9C	E4 BA 9C	4E9C
𝄞	D834 DD1E	F0 9D 84 9E	N/A

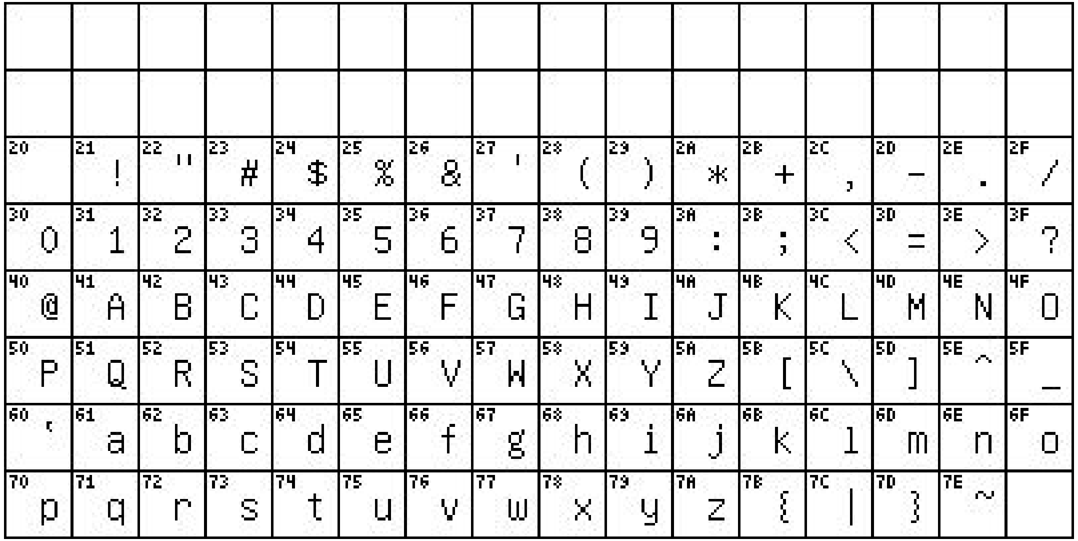


character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

ASCII

Abbreviated from American Standard Code for Information Interchange, is a character encoding standard. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.

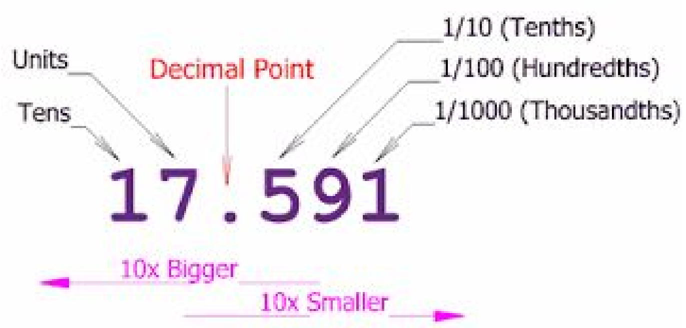
Supports the least Chars.

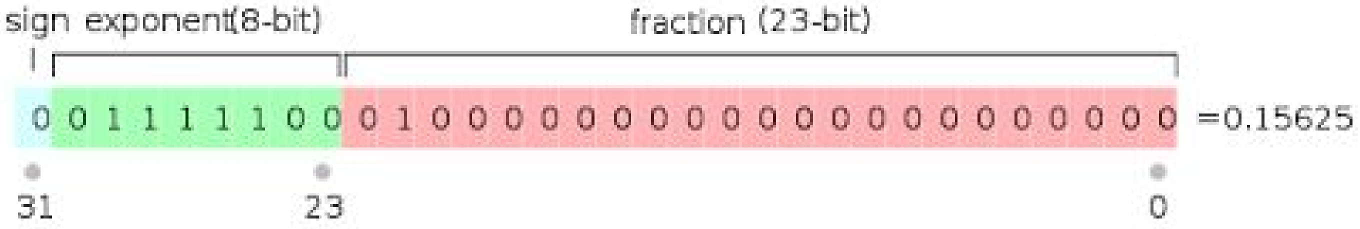


20	21 !	22 "	23 #	24 $	25 %	26 &	27 '	28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7	38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G	48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W	58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g	68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w	78 x	79 y	7A z	7B {	7C |	7D }	7E ~	

Floating point numbers

Floating-point arithmetic is arithmetic using formulaic representation of real numbers as an approximation so as to support a trade-off between range and precision.

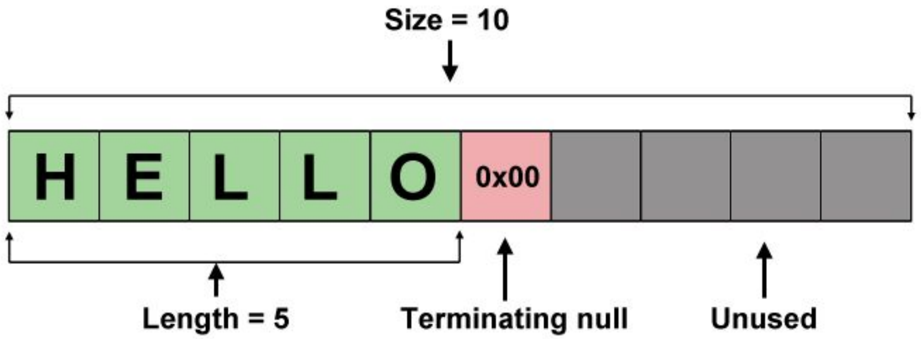




Alphanumeric Strings

a string is traditionally a sequence of characters, is often implemented as an array data structure of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding (hint: unicode and ascii).

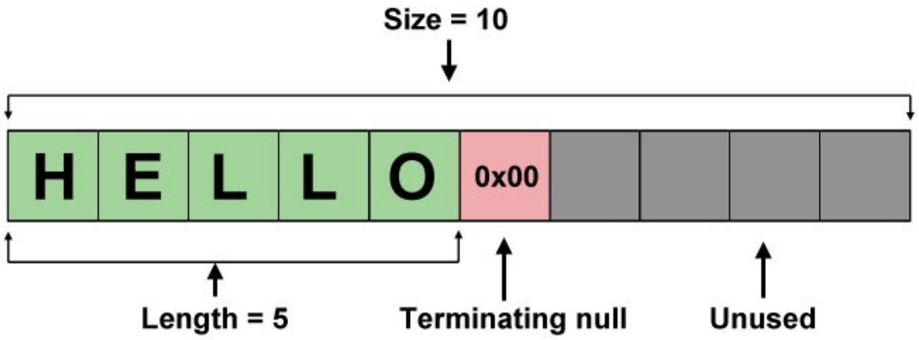
Usually are null-terminated meaning the end of the string is denoted by a special terminating character, null byte (NULL), which is 00000000 (all zeroed out byte)



Reference

a reference is a value that enables a program to indirectly access a particular location in the computer's memory or in some other storage device.

Usually starting at a base address like 0x00000000 and moving up by the size of the data unit.



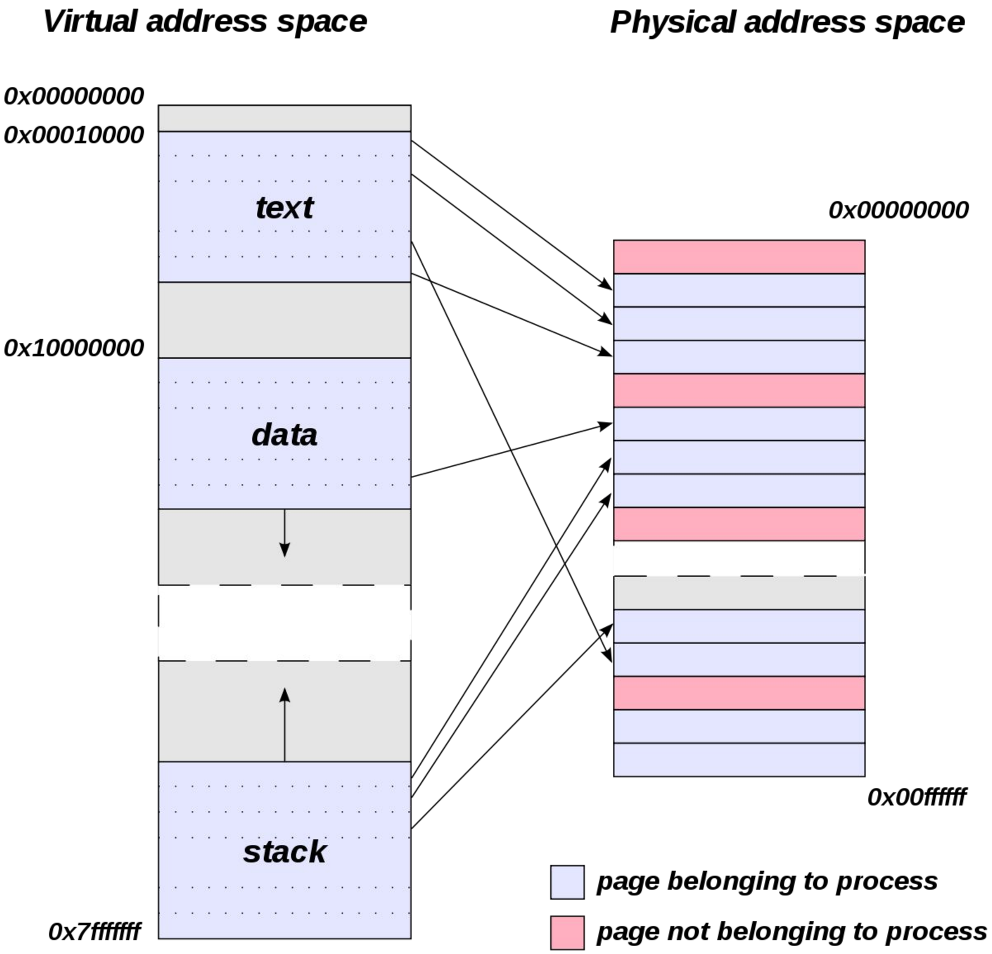
In the above example the ‘E’ would be at 0x00000002 that is 2 byte away from the offset (each char is 2 bytes, so we could assume they might be using UTF-16)

Computer		Programmers		
Address	Content	Name	Type	Value
90000000	00	sum	int (4 bytes)	000000FF(255_{10})
90000001	00			
90000002	00			
90000003	FF			
90000004	FF	age	short (2 bytes)	FFFF(-1_{10})
90000005	FF			
90000006	1F	averge	double (8 bytes)	1FFFFFFFFFFFFFFF ($4.45015E\text{-}308_{10}$)
90000007	FF			
90000008	FF			
90000009	FF			
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			
9000000E	90	ptrSum	int* (4 bytes)	90000000
9000000F	00			
90000010	00			
90000011	00			

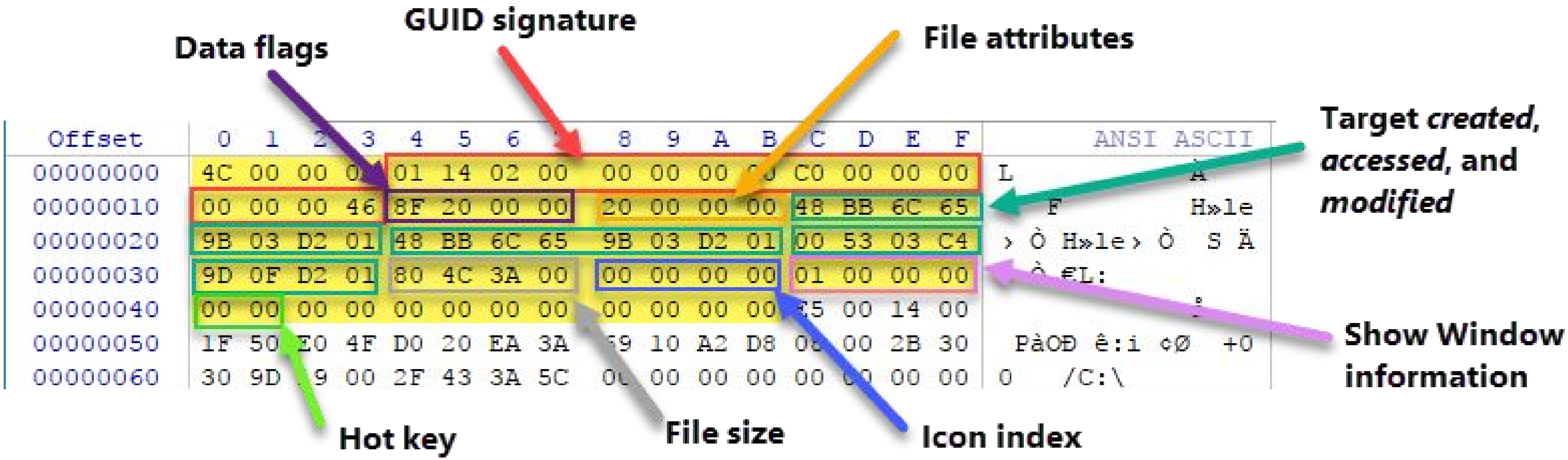
Note: All numbers in hexadecimal

WHY?

Computers think in numbers!



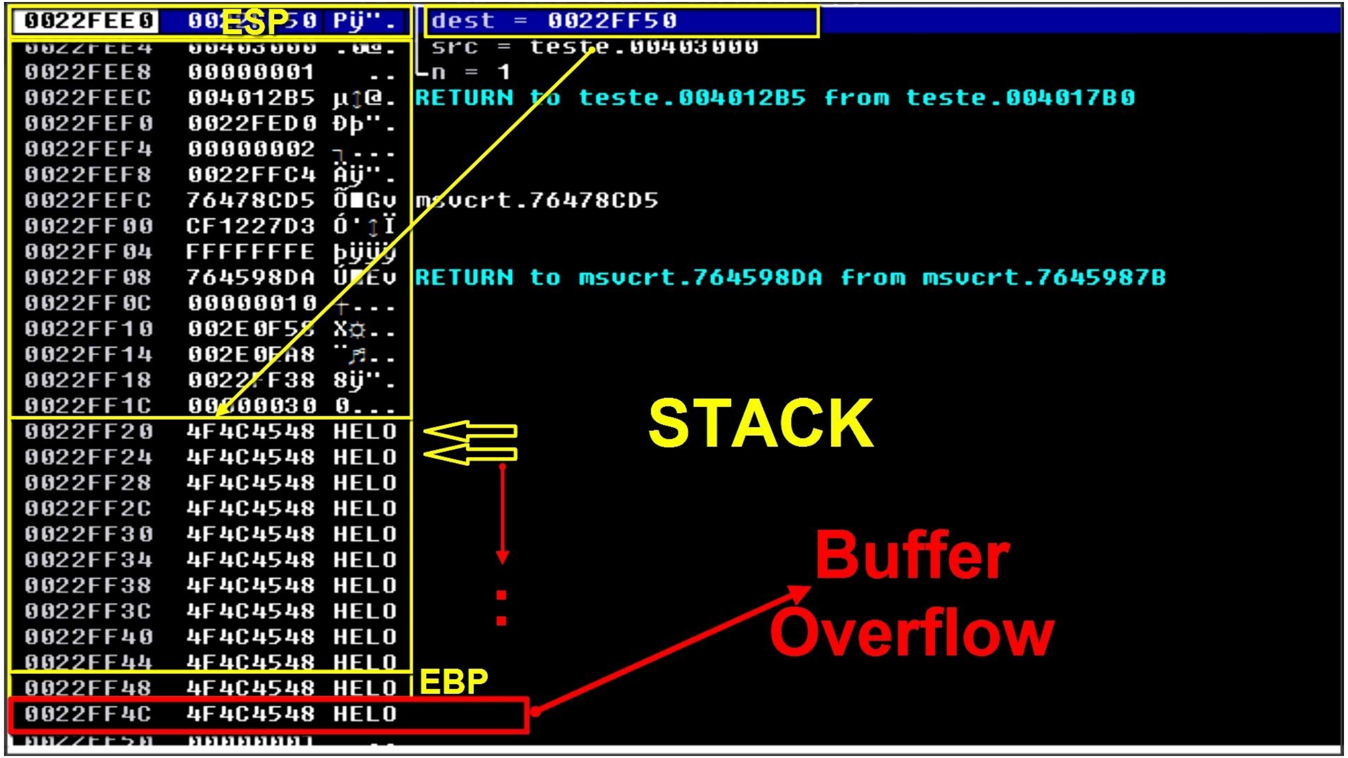
Computers store data in numbers!



```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   ANSI ASCII
00000000  4C 00 00 00 01 14 02 00 00 00 00 00 C0 00 00 00  L         À
00000010  00 00 00 46 8F 20 00 00 20 00 00 00 48 BB 6C 65     F     H»le
00000020  9B 03 D2 01 48 BB 6C 65 9B 03 D2 01 00 53 03 C4  › Ò H»le› Ò  S Ä
00000030  9D 0F D2 01 80 4C 3A 00 00 00 00 00 01 00 00 00   Ò €L:
00000040  00 00 00 00 00 00 00 00 00 00 00 00 C5 00 14 00              Å
00000050  1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30   PàOÐ ê:i ¢Ø  +0
00000060  30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00  0   /C:\
```

Character	ASCII	Decimal	Hex	Binary
L (ASCII)	A	76	4C	01001100

So hackers (both good and bad) need to know how to use these numbers to their advantage!



```
0022FEE0   0022FF50 Pÿ".   dest = 0022FF50
0022FEE4   00403000 .0@.   src = teste.00403000
0022FEE8   00000001   ..   └n = 1
0022FEEC   004012B5 µ↕@.  RETURN to teste.004012B5 from teste.004017B0
0022FEF0   0022FED0 Ðþ".
0022FEF4   00000002 ┐...
0022FEF8   0022FFC4 Äÿ".
0022FEFC   76478CD5 Õ■Gv  msvcrt.76478CD5
0022FF00   CF1227D3 Ó'↕Ï
0022FF04   FFFFFFFE þÿÿÿ
0022FF08   764598DA Ú■Ev  RETURN to msvcrt.764598DA from msvcrt.7645987B
0022FF0C   00000010 ┼...
0022FF10   002E0F58 X☼..
0022FF14   002E0EA8 ¨♬..
0022FF18   0022FF38 8ÿ".
0022FF1C   00000030 0...
0022FF20   4F4C4548 HELO
0022FF24   4F4C4548 HELO
0022FF28   4F4C4548 HELO
0022FF2C   4F4C4548 HELO
0022FF30   4F4C4548 HELO
0022FF34   4F4C4548 HELO
0022FF38   4F4C4548 HELO
0022FF3C   4F4C4548 HELO
0022FF40   4F4C4548 HELO
0022FF44   4F4C4548 HELO
0022FF48   4F4C4548 HELO
0022FF4C   4F4C4548 HELO
0022FF50   00000001   ..
```