Storing Data with Computers

How We Use Numbers

Everything is a power of 10!

Example: 181

How We Use Numbers

Everything is a power of 10!

Example: 181

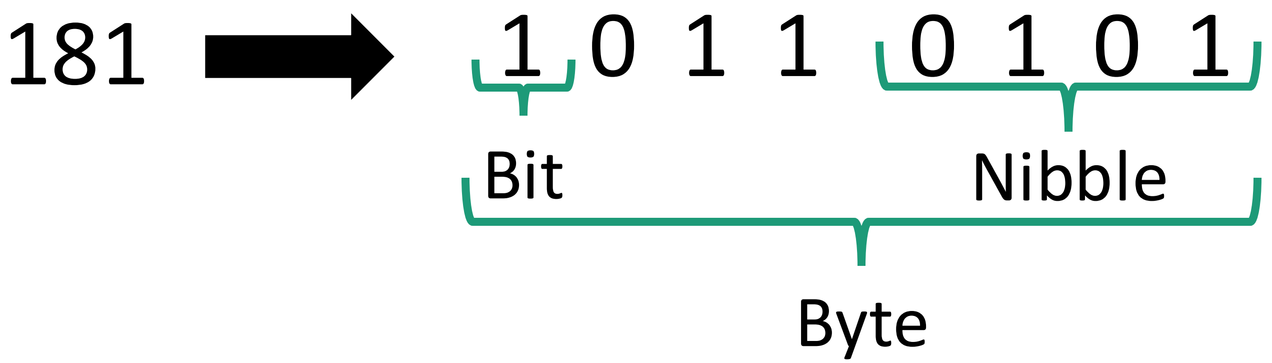
10^3	10^2	10^1	10^0
0	1	8	1

1000*0 + 100 * 1 + 10*8 + 1*1 = 181

How Computers Store Information

Everything is stored in ***binary*** as a series of 1’s and 0’s.

With only two values, this means everything is a power of two!



181

1 0 1 1 0 1 0 1

Bit

Nibble

Byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	0	1	0	1

128*1 + 64*0 + 32*1 + 16*1 + 8*0 + 4*1 + 2*0 + 1*1 = 181

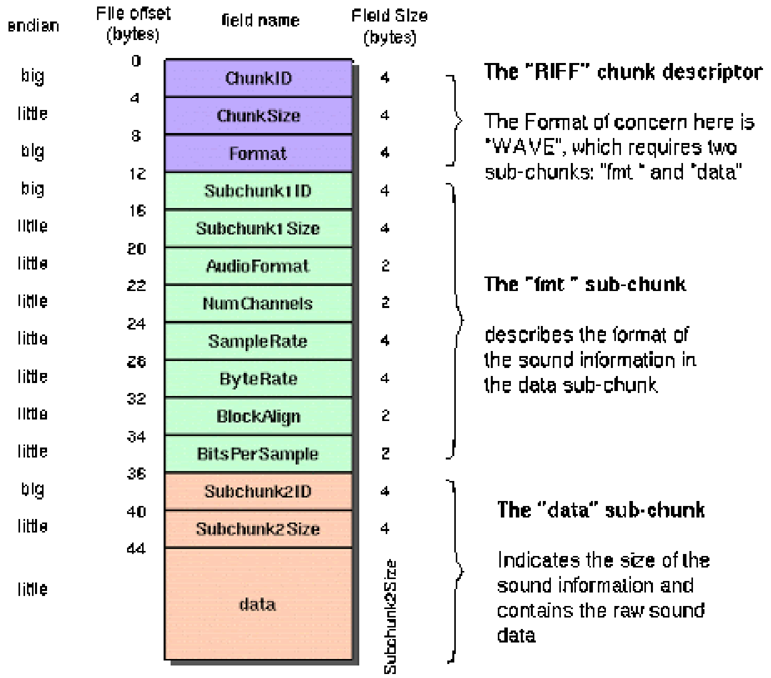
Storing Complex Data

Storing text and other more complex information requires an ***encoding*** format to describe the data in binary/numerical representation.

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

WAV Audio Format



endian	File offset (bytes)	field name	Field Size (bytes)
big	0	ChunkID	4
little	4	ChunkSize	4
big	8	Format	4
big	12	Subchunk1ID	4
little	16	Subchunk1 Size	4
little	20	AudioFormat	2
little	22	NumChannels	2
little	24	SampleRate	4
little	26	ByteRate	4
little	32	BlockAlign	2
little	34	BitsPerSample	2
big	36	Subchunk2ID	4
little	40	Subchunk2Size	4
little	44	data	Subchunk2Size

The "RIFF" chunk descriptor

The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"

The "fmt " sub-chunk

describes the format of the sound information in the data sub-chunk

The "data" sub-chunk

Indicates the size of the sound information and contains the raw sound data

Example

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- Let’s spell the word “ACE” in binary (all capital letters)
- First convert the letter to the decimal value
  - A = 65
- Now convert 65 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0

Example

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- Let’s spell the word “ACE” in binary
- First convert the letter to the decimal value
  - A = 65
- Now convert 65 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0

2^7 = 128

That is far too large.

Leave it zero.

Example

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- Let’s spell the word “ACE” in binary
- First convert the letter to the decimal value
  - A = 65
- Now convert 65 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	0	0	0

2^6 = 64
That is less than or equal to 65.
Let’s mark this with a 1.

Example

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- Let’s spell the word “ACE” in binary
- First convert the letter to the decimal value
    - A = 65
- Now convert 65 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	0	0	1

All we need now is a 1 (65 – 64 = 1).
Let’s mark the 2^0 position with a 1.

Example

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- Let's spell the word "ACE" in binary
- First convert the letter to the decimal value
  - A = 65
- Now convert 65 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	0	0	1

$$2^7(0) + 2^6(1) + 2^5(0) + 2^4(0) + 2^3(0) + 2^2(0) + 2^1(0) + 2^0(1) = 65$$

You Try!

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- A = 65 = 01000001
- Try to convert capital C and E to binary on your own!

You Try!

ASCII

Decimal	Character
65	A
66	B
67	C
68	D
69	E
70	F
…	…

- A = 65 = 01000001
- Try to convert capital C and E to binary on your own!

C = 67

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	0	1	1

$2^7(0) + 2^6(1) + 2^5(0) + 2^4(0) + 2^3(0) + 2^2(0) + 2^1(1) + 2^0(1) = 67$

E = 69

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	1	0	1

$2^7(0) + 2^6(1) + 2^5(0) + 2^4(0) + 2^3(0) + 2^2(1) + 2^1(0) + 2^0(1) = 69$

Floating Point Numbers

- Floating point numbers are more complicated to store

- The Institute of Electrical and Electronics Engineers (IEEE) have created an encoding format for representing these values
  - The standard is called IEEE 754
  - YOU DO NOT NEED TO KNOW HOW TO DO THIS CONVERSION ☺

- It is not possible to represent all floating-point numbers
  - WHY?
  - The number of possible values between 0 and 1 is infinite and computers have finite storage!

Hexadecimal

- Binary numbers can get quite long
  - Even simple things like integers could use up to 64 bits!

- When binary numbers are presented to people, they often take the form of hexadecimal values as we can represent the same data an abbreviated fashion

- You may have already seen hexadecimal values before as they are very common for the use of color on the web
  - This is red: FF0000

Representing Binary as Hexadecimal Values

- Hexadecimal numbers use the values 0-9 and A-F
  - 0-9 represent the numbers 0-9 (0000 – 1001)
  - A-F represent the numbers 10-15 (1010 – 1111)
  - 16 possible values means we now are dealing in powers of 16

- Each hexadecimal represents a group of four binary digits
  - Ex. 01 1001 1111 or 0001 1001 1111

- We can convert the groups of four to hexadecimal digits and combine them
  - 0001 = 1
  - 1001 = 9
  - 1111 = F (15)

Binary Hexadecimal
0b01 1001 1111 => 0x19F

NOTE: The 0b and 0x prefixes just distinguish
binary and hexadecimal format respectively

Try it yourself!

- Convert decimal 635 to binary
  - 0b1001111011

- Convert decimal 635 to hexadecimal
  - 0b1001111011 = 0x27B

- Convert hexadecimal 0x7C to decimal
  - $7(16^1) + 12(16^0) = 124$