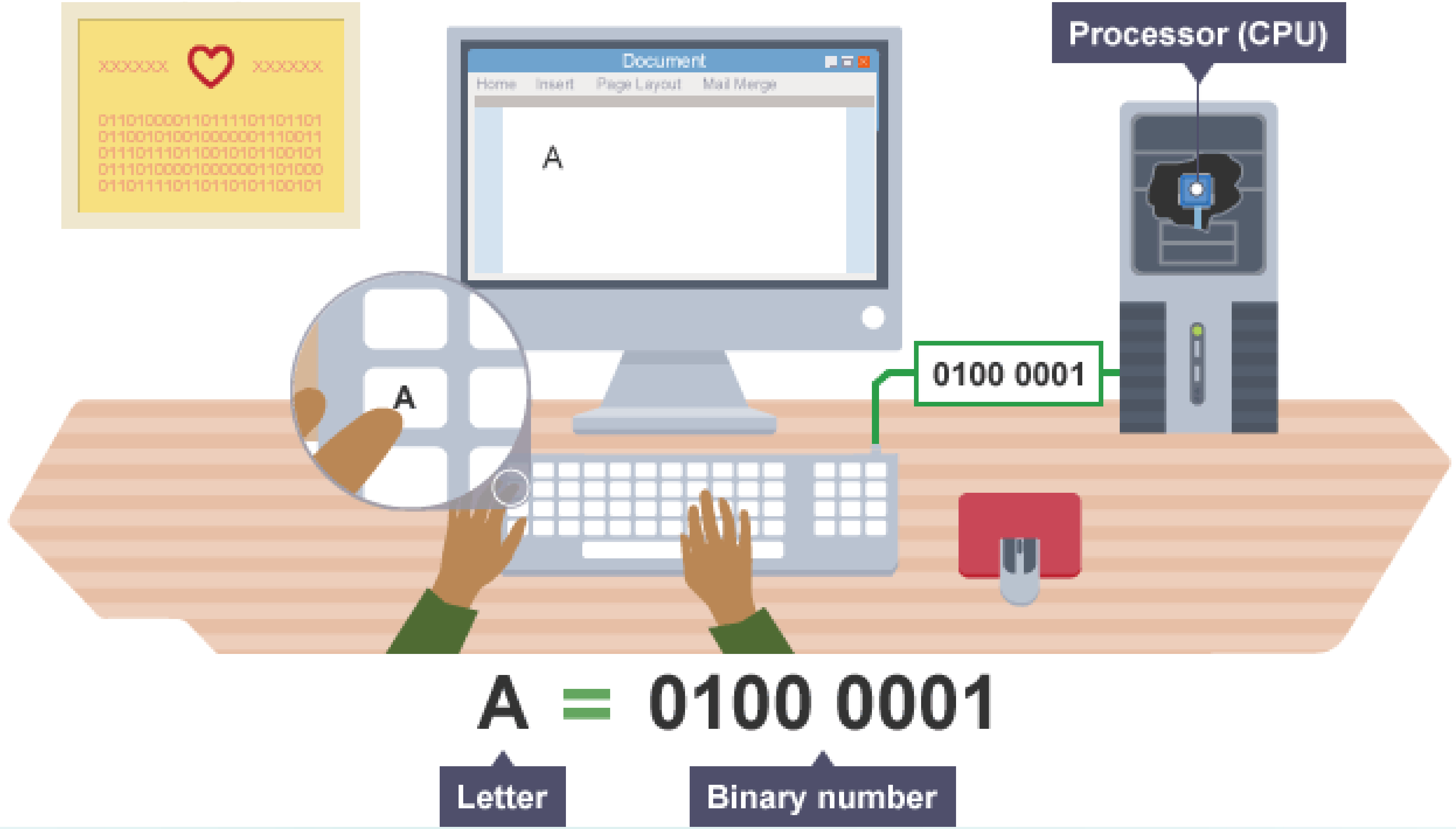
Topic 02
Digital Representation

Outline

1) Number Representation
2) Arithmetic Operations
3) Character Representation

Representing text



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]

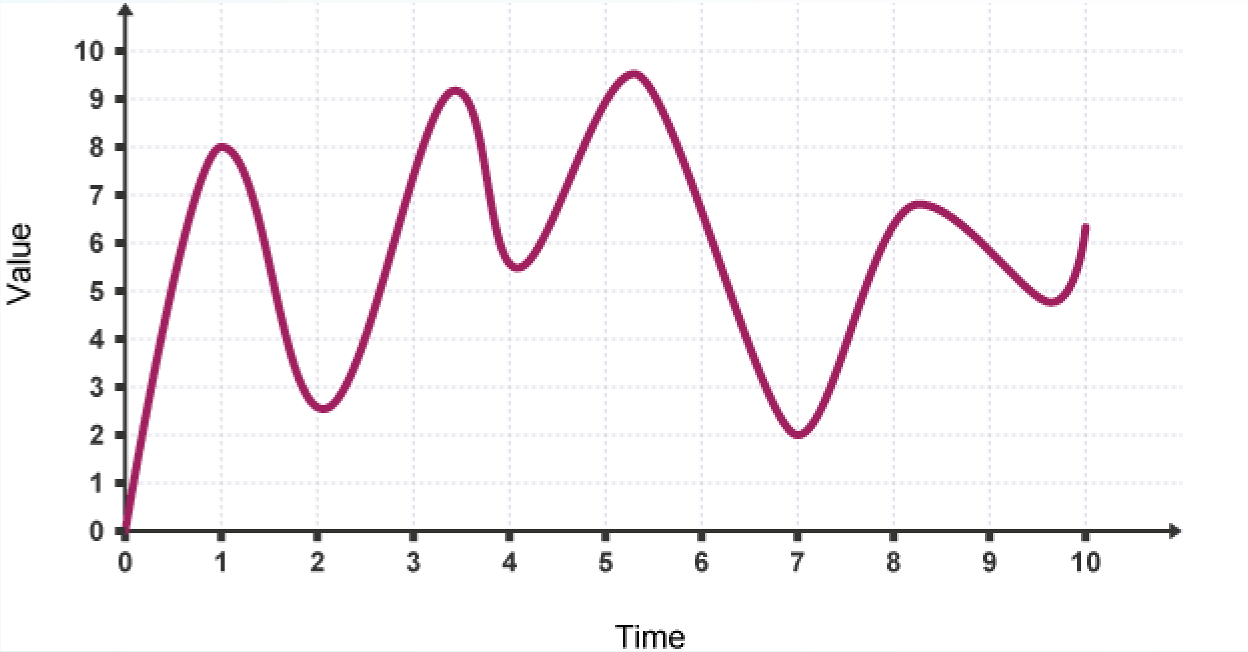
ASCII code can only store 128 characters, which is enough for most words in English but not enough for other languages. If you want to use accents in European languages or larger alphabets such as Cyrillic (the Russian alphabet) and Chinese Mandarin then more characters are needed. Therefore another code, called **Unicode**, was created. This meant that computers could be used by people using different languages.

Chinese Character “我” Unicode u6211

Representing sound

Sound needs to be converted into **binary** for computers to be able to process it. To do this, sound is captured - usually by a microphone - and then converted into a **digital** signal.

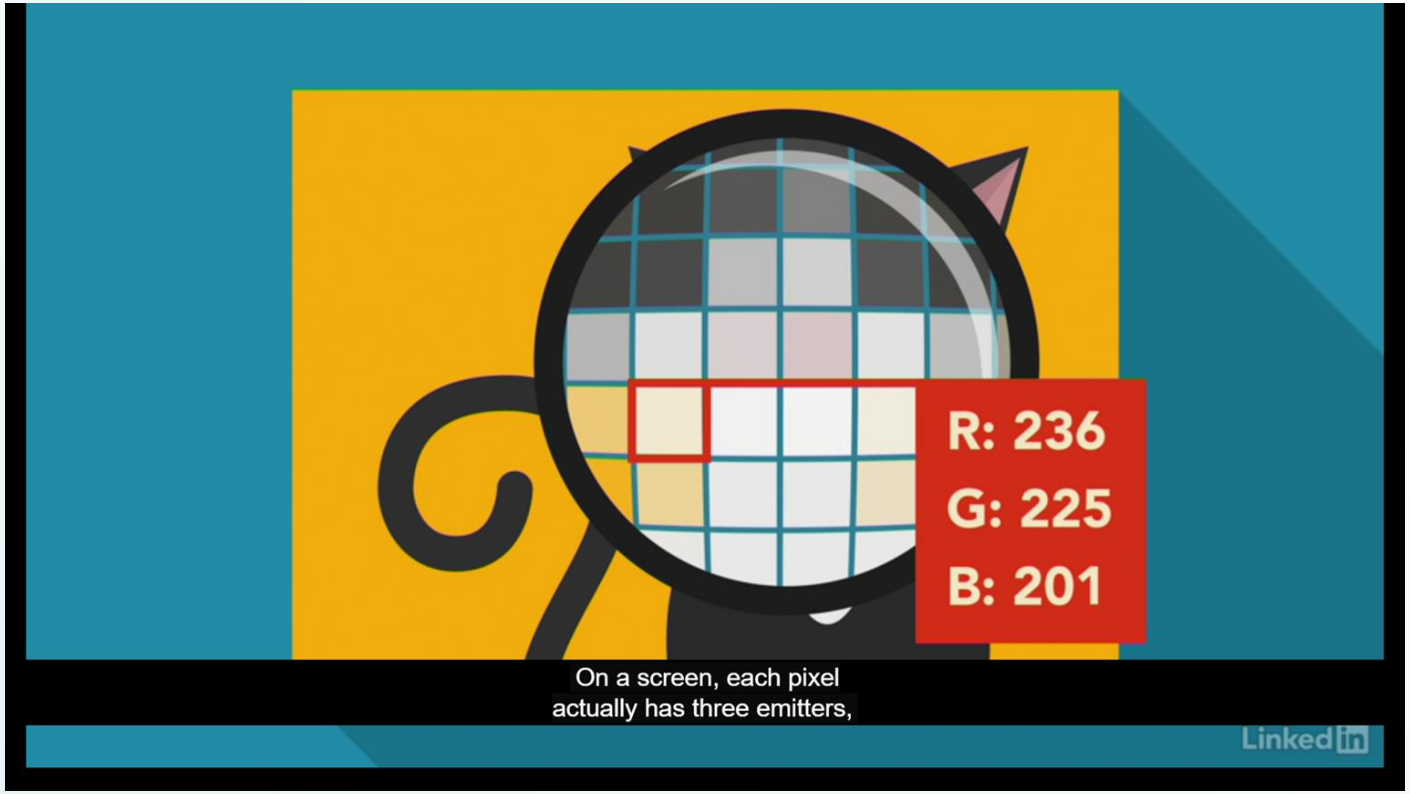
The samples can then be converted to binary. They will be recorded to the nearest whole number.



Time sample	1	2	3	4	5	6	7	8	9	10
Denary	8	3	7	6	9	7	2	6	6	6
Binary	1000	0011	0111	0110	1001	0111	0010	0100	0110	0110

Encoding Images

When you have an image, it is broken up into individual pixels.



Color Picker
https://www.w3schools.com/colors/colors_picker.asp

Representing number

The most natural way to represent a number in a computer system is by a string of bits, call binary number.

Number Representation - Decimal

Radix	10	10	10	10
Position In	3	2	1	0
Calculate	10^3	10^2	10^1	10^0
Positional Value	1000	100	10	1

1234

	Thousands	Hundreds	Tens	Ones
Positional Value	1000	100	10	1
Decimal Number	1	2	3	4
Calculate	1x1000	2x100	3x10	4x1
Add them	1000	+200	+30	+4
Result	1234			

4321

	Thousands	Hundreds	Tens	Ones
Positional Value	1000	100	10	1
Decimal Number	4	3	2	1
Calculate	4x1000	3x100	2x10	1x1
Add them	4000	+300	+20	+1
Result	4321			

People use the **denary** (or decimal) number system in their day-to-day lives. This system has 10 digits that we can use: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Numbers

In decimal, the place values are 1, 10, 100, 1000, etc – each place value is 10 times bigger than the last. In binary, each place value is 2 times bigger than the last (ie increased by the power of 2). The first few binary place values look like this:

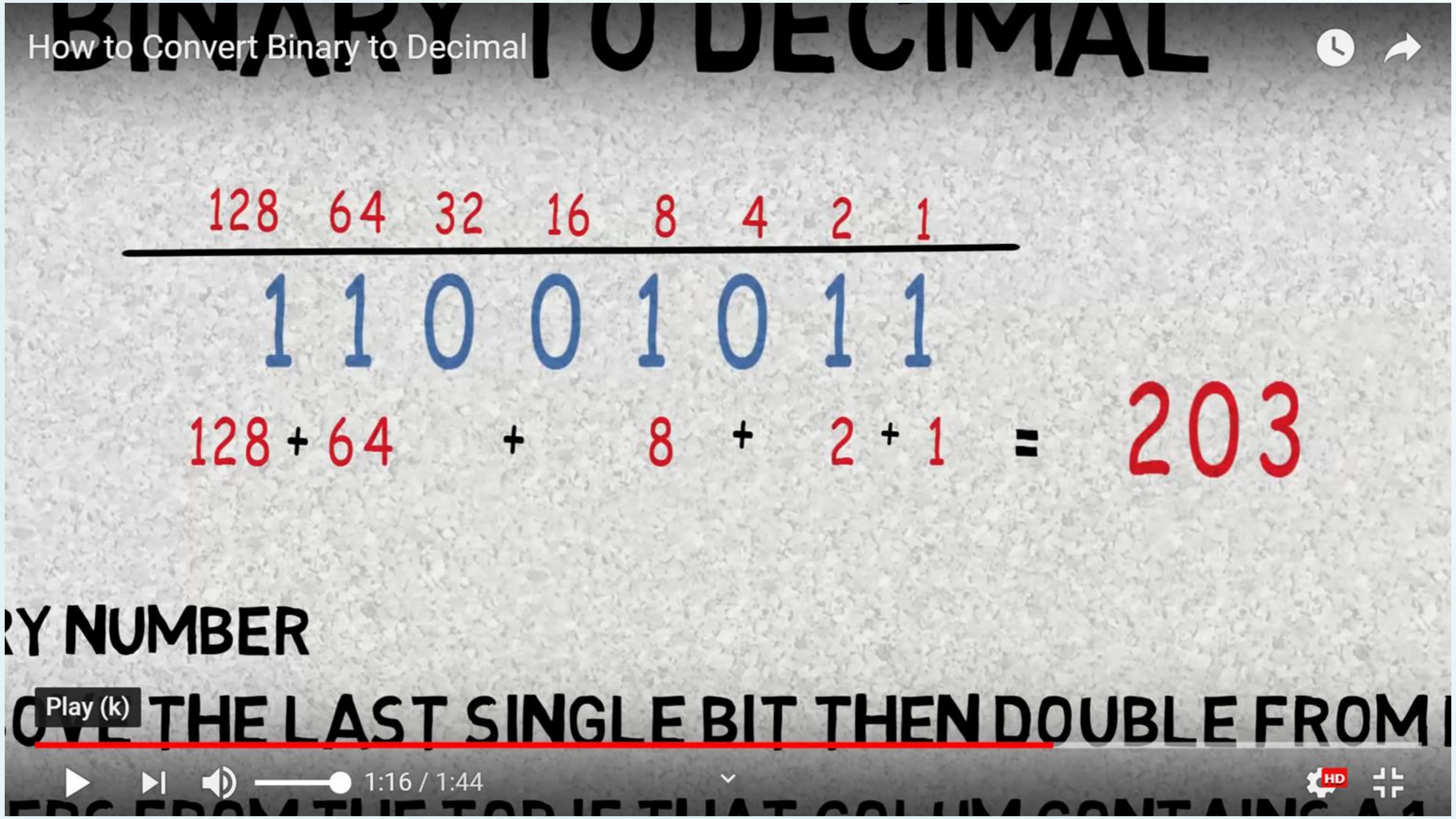
MSB	Binary Digit							LSB
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

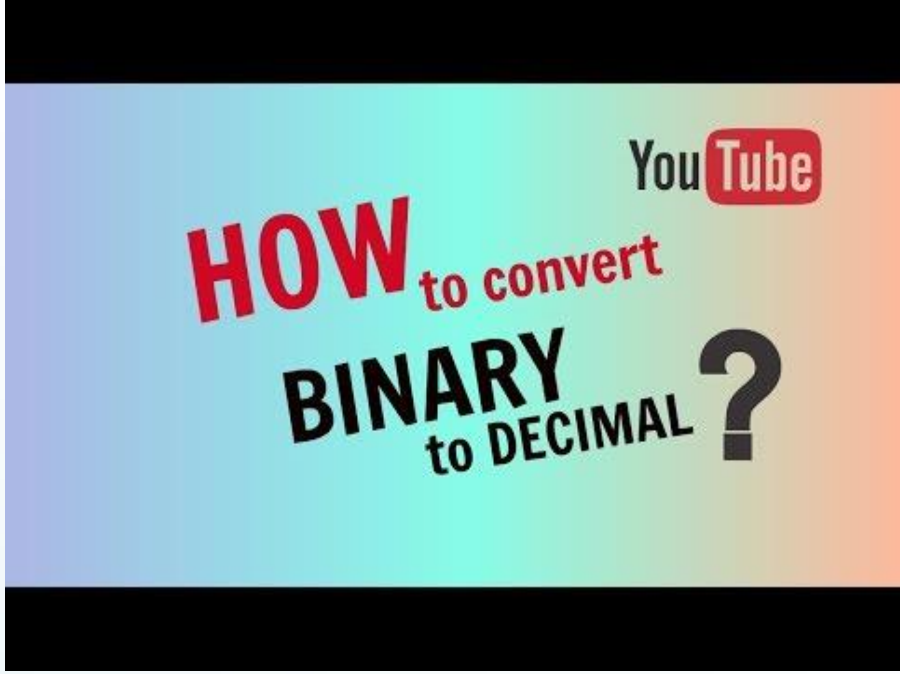
How to convert binary to decimal?

Use the weighted sum method

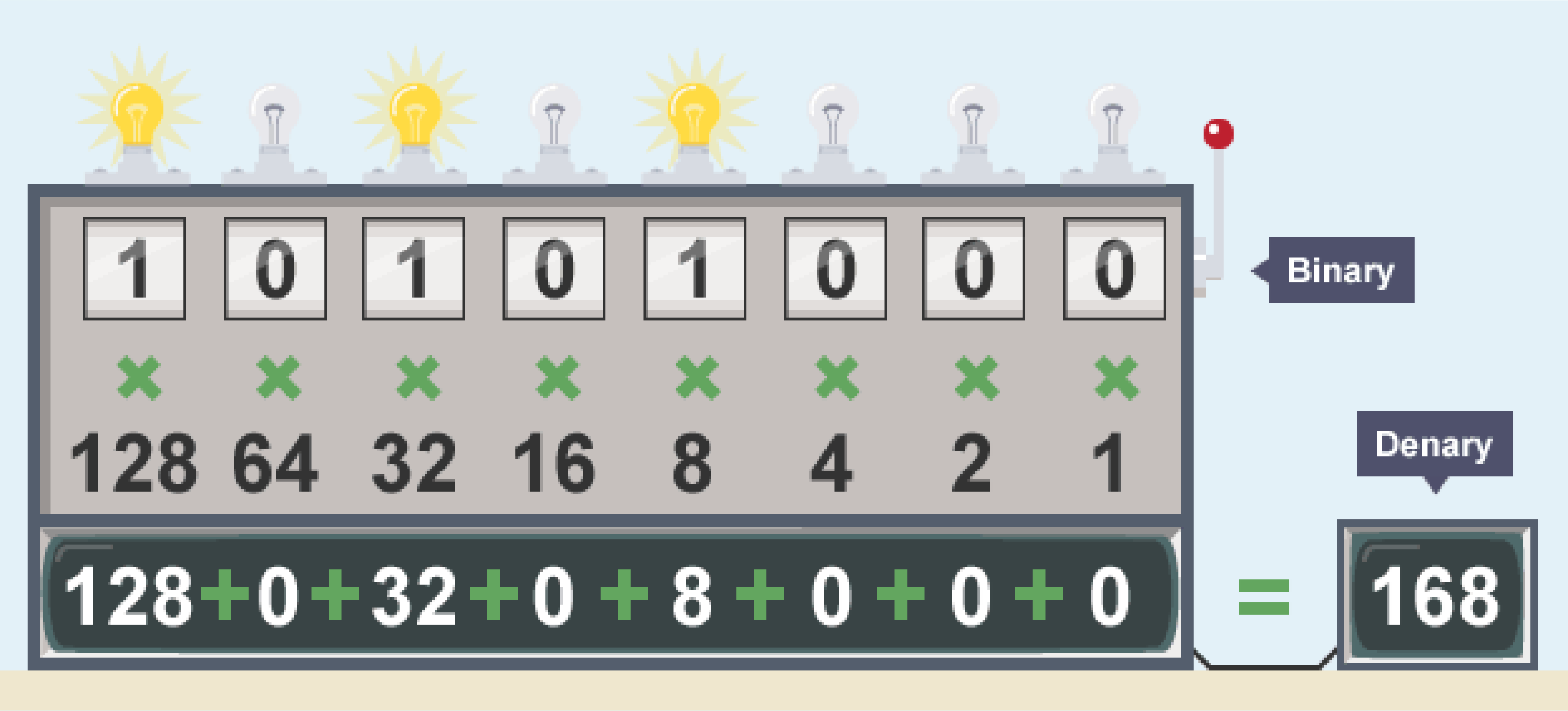
By adding together ALL the decimal number values from right to left at the positions that are represented by a “1” gives us: (256) + (64) + (32) + (4) + (1) = 357_{10} or three hundred and fifty seven as a decimal number.

Decimal Digit Value	256	128	64	32	16	8	4	2	1
Binary Digit Value	1	0	1	1	0	0	1	0	1


How to Convert Binary to Decimal
128 64 32 16 8 4 2 1
1 1 0 0 1 0 1 1
128 + 64 + 8 + 2 + 1 = 203
RY NUMBER
Play (k)
THE LAST SINGLE BIT THEN DOUBLE FROM
1:16 / 1:44


YouTube
HOW to convert
BINARY to DECIMAL ?

Numbers



How to convert decimal numbers to binary?

Let’s try to convert the number 19 to binary:

The procedure is to keep dividing the number by 2, and keeping track of the remainder until we get 0 as a result of the division. Then read the remainder values in reverse order

Division	Quotient	Remainder	Value	
19 / 2	9	1	2^0	← LSB
9 / 2	4	1	2^1	
4 /2	2	0	2^2	
2 / 2	1	0	2^3	
1 / 2	0	1	2^4	← MSB

Answer: 19 is 10011 binary

Number Representation - Unsigned

If there are N bits in the binary number, the range of the number is:
0 .. 2^N-1. Assume 3 bits: 2^3-1 = 8-1=7

000, 001, 010, 011
100, 101, 110, 111

Range for 8 bits? 0..2^8-1 = 0..256-1=0..255

Why unsigned numbers?
Memory address, cluster number (file system) and process identifier (PID).

Number Representation

You will often come across strings of bits, such as:

`0010101010001101 (16 bit)`

`10001001011001111010111010101110 (32 bit)`

They can be difficult for humans to read and write accurately.

To facilitate readability and conserve space when viewed in a monitor or in print, we use the Octal (Base 8) and Hexadecimal (Base 16) system as a shorthand to represent binary.

Number Representation - Hex

Line 09 – Start address

Line 10:

00000002 +2 bytes

[Macro: expanded to many instructions]

```
 3                                        section .data
 4 00000000 48656C6C6F2C20776F-               msg db 'Hello, world!', 0
 4 00000009 726C642100
 5
 6                                        section .text
 7                                            global CMAIN
 8                                        CMAIN:
 9 00000000 89E5                              mov ebp, esp
10 00000002 E802000000EB6AEB00-               PRINT_STRING msg
10 0000000E 0000000538B5C2404-
10 00000010 891D[0E000000]8B1C-
10 00000018 24895C24045B5B9C52-
10 00000021 5150C800000083EC00-
10 0000002A 83E4F083C400E8-
10 00000031 (00000000)C9C80000-
10 ...
11 00000073 E802000000EB6EEB00-               NEWLINE
11 ...
12 000000E8 31C0                              xor eax, eax
13 000000EA C3                                ret
```

Number Representation (Hexadecimal to Decimal)

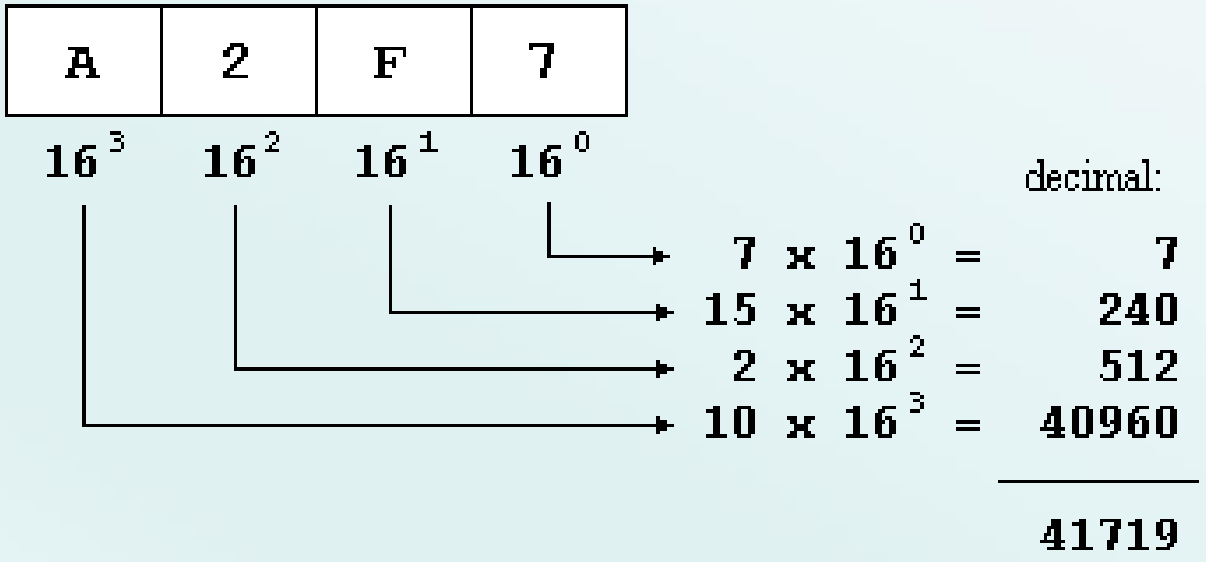
Hexadecimal (Base 16):

$H = h_3\ h_2\ h_1\ h_0$ (h = hexadecimal ranging from 0 to 9, A to F)

If $h_i = 1$, then 16^i is added into the sum that determines the value of H.

For values > 9, we use A,B,C,D,E,F to represent digits up to 15.

Let's assume that H = A 2 F 7



Number Representation

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Binary number to hexadecimal conversion

Bin = 0011 0101 0111 0011

Hex = 3573

Denoting Hexadecimal

1. Place 0x before the hex numeral: 0x3573
2. Append a lowercase letter h after the numeral: 3573h
3. Use the 16 subscript: 3573_{16}
4. Specify base 16: 3573 (base 16)

Number Representation

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Binary number to hexadecimal conversion

Bin = 11 0010 1010 1000

Hex = ?

If there are not enough MSB digits, pad with 0s to form group of 4 binary bits.

Hex = 0x32A8

Number Representation

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Binary number to hexadecimal conversion

$10001001011001111010111010101110_b$

Hex = ?

Hex = 0x8967AEAE

Number Representation

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Hexadecimal to binary number conversion

Hex = F9C7AEAE

Bin = ?

$1111\ 1001\ 1100\ 0111\ 1010\ 1110\ 1010\ 1110_b$

Notation for binary and hexadecimal number (summary)

Mathematician's way

Binary (base 2)

10101111_2

Hexadecimal (base 16)

AF_{16}

Decimal (base 10)

175

Number Representation - Signed

For signed integers, the leftmost bit is used to indicate the sign:

- 0 for positive
- 1 for negative

There are three ways to represent signed integers:

a) Sign and magnitude
b) 1’s complement
c) 2’s complement

Number Representation - Signed

B	Values represented		
$b_3 b_2 b_1 b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+7	+7	+7
0 1 1 0	+6	+6	+6
0 1 0 1	+5	+5	+5
0 1 0 0	+4	+4	+4
0 0 1 1	+3	+3	+3
0 0 1 0	+2	+2	+2
0 0 0 1	+1	+1	+1
0 0 0 0	+0	+0	+0
1 0 0 0	−0	−7	−8
1 0 0 1	−1	−6	−7
1 0 1 0	−2	−5	−6
1 0 1 1	−3	−4	−5
1 1 0 0	−4	−3	−4
1 1 0 1	−5	−2	−3
1 1 1 0	−6	−1	−2
1 1 1 1	−7	−0	−1

Number Representation - Signed

Positive values have identical representations in all systems.

Negative numbers have different representations

1) Sign-and-magnitude: negative values are represented by changing the most significant bit (b_3).
2) 1's-complement: negative values are obtained by complementing each bit of the corresponding positive number.
3) 2's-complement: obtain by forming bit complement of that number, then add 1.

2’s-complement integers

2’s-complement representation is used in current computers

Consider a four-bit signed integer example, where the value +5 is represented as:

0 1 0 1

To form the value -5, complement all bits of

0 1 0 1 to obtain

1 0 1 0 (1’s-complement)

and then add 1 to obtain

1 0 1 1

2’s-Complement

2’s-complement system is the most efficient way to carry out addition and subtraction operations.

It is the most often used in modern computers

Using 1’s complement	Using 2’s complement
1100 (-3)	1101 (-3)
+ 1101 (-2) +	1110 (-2)
-----------------	-------------------
1001 (-6)	1011 (-5)

Overflow check needed   Ignore overflow

Addition

There are four rules that need to be followed when adding two binary numbers. These are:

- 0 + 0 = 0
- 1 + 0 = 1
- 1 + 1 = 10 (binary for 2)
- 1 + 1 + 1 = 11 (binary for 3)

Addition

Two examples of adding four-bit numbers:

0 0 0 1	+1	0 1 0 0	+4
+ 0 1 $0_1$1	+ +5	+ 1 0 1 0	+ -6
-------------	--------	-------------	------
0 1 1 0	+6	1 1 1 0	-2

Addition

(a)

$$\begin{array}{r} 0010 \\ +\ 0011 \\ \hline 0101 \end{array} \qquad \begin{array}{r} (+2) \\ (+3) \\ \hline (+5) \end{array}$$

(b)

$$\begin{array}{r} 0100 \\ +\ 1010 \\ \hline 1110 \end{array} \qquad \begin{array}{r} (+4) \\ (-6) \\ \hline (-2) \end{array}$$

(c)

$$\begin{array}{r} 1011 \\ +\ 1110 \\ \hline 1001 \end{array} \qquad \begin{array}{r} (-5) \\ (-2) \\ \hline (-7) \end{array}$$

(d)

$$\begin{array}{r} 0111 \\ +\ 1101 \\ \hline 0100 \end{array} \qquad \begin{array}{r} (+7) \\ (-3) \\ \hline (+4) \end{array}$$

Subtraction

Form the 2's-complement of the subtrahend (the bottom value) and then perform normal addition

This operation is done in exactly the same manner for both positive and negative numbers

```
      1 1 1 0           -2                    1 1 1 0
  -   1 0 1 1     -     -5         ⇒        + 0 1 0 1
  -----------        ------                 -----------
                        +3                    0 0 1 1
```

Subtraction

(e)
$$\begin{array}{r} 1101 \\ -\ 1001 \\ \hline \end{array} \quad \begin{array}{r} (-3) \\ (-7) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 1101 \\ +\ 0111 \\ \hline 0100 \end{array} \quad \begin{array}{r} \\ \\ \hline (+4) \end{array}$$

(f)
$$\begin{array}{r} 0010 \\ -\ 0100 \\ \hline \end{array} \quad \begin{array}{r} (+2) \\ (+4) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 0010 \\ +\ 1100 \\ \hline 1110 \end{array} \quad \begin{array}{r} \\ \\ \hline (-2) \end{array}$$

(g)
$$\begin{array}{r} 0110 \\ -\ 0011 \\ \hline \end{array} \quad \begin{array}{r} (+6) \\ (+3) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 0110 \\ +\ 1101 \\ \hline 0011 \end{array} \quad \begin{array}{r} \\ \\ \hline (+3) \end{array}$$

Subtraction

(h)

$$\begin{array}{rr} & 1001 \\ - & 1011 \\ \hline \end{array} \quad \begin{array}{r} (-7) \\ (-5) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{rr} & 1001 \\ + & 0101 \\ \hline & 1110 \end{array} \quad \begin{array}{r} \\ \\ \hline (-2) \end{array}$$

(i)

$$\begin{array}{rr} & 1001 \\ - & 0001 \\ \hline \end{array} \quad \begin{array}{r} (-7) \\ (+1) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{rr} & 1001 \\ + & 1111 \\ \hline & 1000 \end{array} \quad \begin{array}{r} \\ \\ \hline (-8) \end{array}$$

(j)

$$\begin{array}{rr} & 0010 \\ - & 1101 \\ \hline \end{array} \quad \begin{array}{r} (+2) \\ (-3) \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{rr} & 0010 \\ + & 0011 \\ \hline & 0101 \end{array} \quad \begin{array}{r} \\ \\ \hline (+5) \end{array}$$

Arithmetic

What can you observe about signed arithmetic?

0 1 1 0	+6	1 1 1 0	-2
+ $0_1$1 0 0	+ +4	$_1$1 0 0 1	+ -7
------------	------	----------	------
1 0 1 0	+10	0 1 1 1	-9

The answers are incorrect! It does not fit in the number range (Overflow)

For signed integers, the leftmost bit is used to indicate the sign:

- 0 for positive
- 1 for negative
- Range: -2^{n-1} to $+2^{n-1}-1$
- 4 bits: -8 to 7 (2^3 to 2^3-1) , 8 bits: -128 to 127 (2^7 to 2^7-1)

Powers of 2 Table - Vaughn's Summaries.mhtml

Overflow

Overflow occurs when the answer does not fit in the number range

	Binary	Decimal
	0 1 1 0	+6
+	$0_{1}1\ 0\ 0$	+ +4
	------------	------
	1 0 1 0	+10

	Binary	Decimal
	1 1 1 0	-2
	$_{1}1\ 0\ 0\ 1$	+ -7
	----------	------
	0 1 1 1	-9

The answers are incorrect! Why?

For signed integers, the leftmost bit is used to indicate the sign:

- 0 for positive
- 1 for negative

Range: -2^{n-1} to $+2^{n-1}-1$

4 bits: -8 to 7 (-2^3 to 2^3-1) , 8 bits: -128 to 127 (-2^7 to 2^7-1)

Powers of 2 Table - Vaughn's Summaries.mhtml

Overflow (Addition)

If 2 Two's Complement numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs **if and only if** the result has the opposite sign. Overflow never occurs when adding operands with different signs.
Adding two positive numbers must give a positive result
Adding two negative numbers must give a negative result

Overflow occurs if:
(+A) + (+B) = −C
(−A) + (−B) = +C

Example: Using 4-bit Two's Complement numbers ($-8 \leq x \leq +7$)

```
 (-7)    1001
+(-6)    1010
----------------
(-13) 1 0011 = 3: Overflow (largest -ve number is -8)
```

Overflow (Subtraction)

If 2 Two's Complement numbers are subtracted, and their signs are different, then overflow occurs **if and only if** the result has the same sign as the subtrahend (what is being subtracted).

Overflow occurs if
(+A) − (−B) = −C
(−A) − (+B) = +C
Example: Using 4-bit Two's Complement numbers (−8 ≤ x ≤ +7)

Subtract −6 from +7

```
 (+7) 0111                  0111
–(–6) 1010 -> Negate -> +0110
      ----                  ----
  13                        1101 = 7–(-6)= –3: Overflow
```

Overflow (Check)

Study the following code:

```
; check for overflow
    xor al, al
    mov al, 01111111b
    add al, 00000001b
    jo ovf_num
    PRINT_DEC 1, al
ovf_num:
    PRINT_STRING "Range for 1 byte: -128 to 127; 127 + 1 will cause overflow"
    NEWLINE
    PRINT_STRING "Overflow"
    NEWLINE
    PRINT_STRING " OK, Unsigned: "
    PRINT_UDEC 1, al
    NEWLINE
    PRINT_STRING "NOK,   Signed: "
    PRINT_DEC 1, al
```

As programmer, you decide whether a number is
SIGNED
or
UNSIGNED !!

Case Study: A bug, A crash

It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to **stuff a 64-bit number into a 16-bit space.**

One bug, one crash. Of all the careless lines of code recorded in the annals of computer science, this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.

Source: https://around.com/ariane.html

Sign extension

Replicate the sign bit to extend 4-bit signed integers to 8-bit signed integers

0 1 0 1 → 0 0 0 0 0 1 0 1

1 1 1 0 → 1 1 1 1 1 1 1 0

Character Representation

American Standard Code for Information Interchange (ASCII)
Uses 7-bit codes

Examples:

character	code
A	1 0 0 0 0 0 1
7	0 1 1 0 1 1 1
+	0 1 0 1 0 1 1

Summary

1) Number Representation
2) Arithmetic Operations - Overflow
3) Character Representation