

# Computers

# Data Representation

## Exercises

*Grau en Ciència i Enginyeria de Dades*

---

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2021-2022 Q2

# Creative Commons License

---

This work is under a Creative Commons Attribution 4.0 Unported License



The details of this license are publicly available at <https://creativecommons.org/licenses/by-nc-nd/4.0>

# Binary arithmetic

---

- Binary addition

$$\begin{array}{r} 10110010 \\ + 01011001 \\ \hline \end{array}$$

Carry bit

The diagram illustrates the binary addition of two 8-bit numbers. The first number is 10110010 and the second is 01011001. A plus sign is placed between them. Below the plus sign is an equals sign, followed by a row of eight empty boxes representing the result. To the left of this row is a solid blue box, which is labeled 'Carry bit' below it.

# Binary arithmetic

---

- Substraction

$$\begin{array}{r} 10110010 \\ - 01011001 \\ \hline \end{array}$$

Carry bit

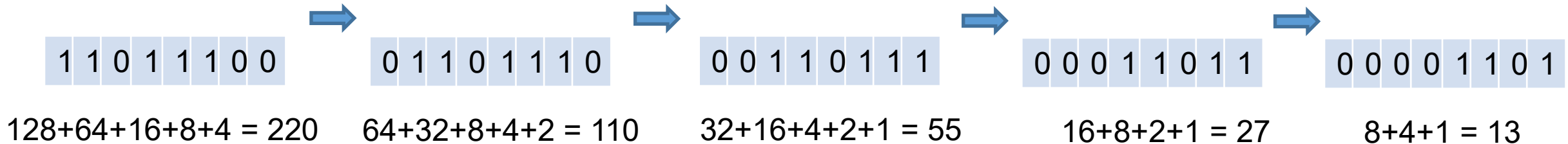
The diagram shows a binary subtraction problem. The first number is 10110010 and the second is 01011001. A minus sign is placed between them. Below the numbers is an equals sign, followed by a row of 9 empty boxes representing the result. To the left of the first box in this row is a blue square labeled 'Carry bit'.

- do you see a property shared between the two numbers?

# Binary arithmetic

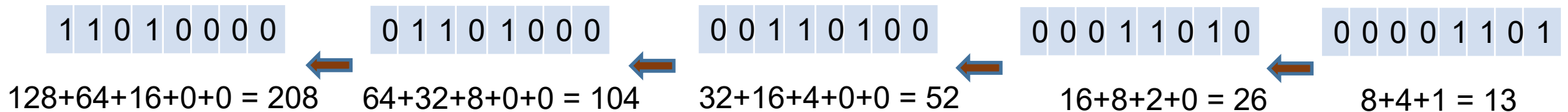
- Division /2 - shift right

- Right-most bit is lost



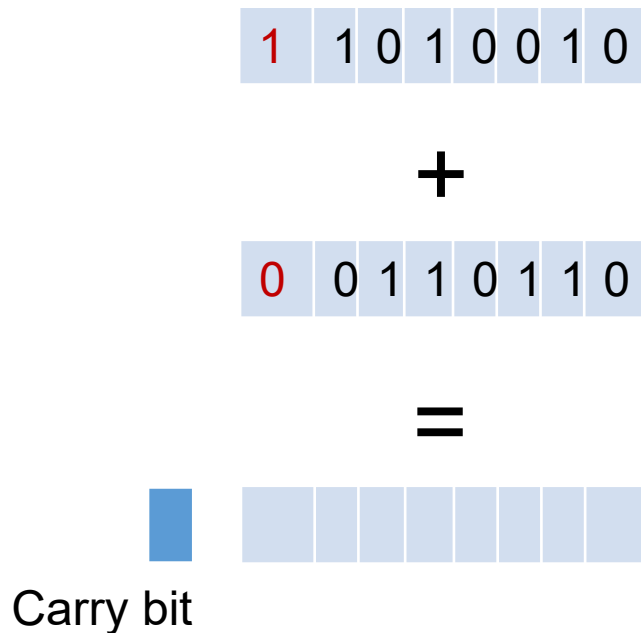
- Multiplication \*2 - shift left

- Left-most bit gets into the carry



# Binary arithmetic

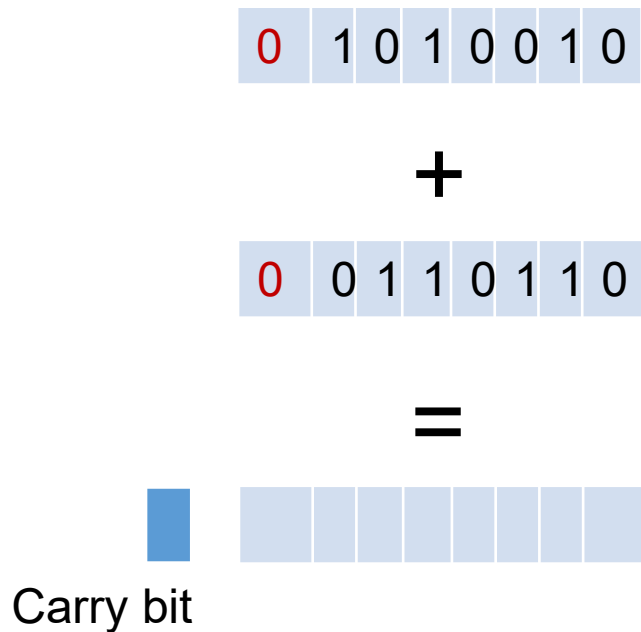
- Signed numbers ... most significant bit represents the sign



Into sign	Into carry	Overflow
0	0	0
0	1	1
1	0	1
1	1	0

# Binary arithmetic

- Signed numbers ... most significant bit represents the sign



Into sign	Into carry	Overflow
0	0	0
0	1	1
1	0	1
1	1	0

# Binary logical operations

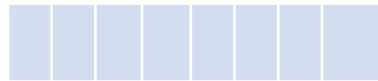
- And

1 0 1 1 0 0 1 0

AND

0 1 0 1 1 0 0 1

=

■ 

Carry bit

■

Overflow

Src 0	Src 1	AND
0	0	0
0	1	0
1	0	0
1	1	1

\*this operation may or may not modify the carry and overflow bits depending on the particular processor brand (Intel, ARM, IBM...)



# Binary logical operations

- Or

1 0 1 1 0 0 1 0

OR

0 1 0 1 1 0 0 1

=

1 1 1 1 1 0 0 1

Carry bit

Overflow

Src 0	Src 1	OR
0	0	0
0	1	1
1	0	1
1	1	1

\*this operation may or may not modify the carry and overflow bits depending on the particular processor brand (Intel, ARM, IBM...)

# Binary logical operations

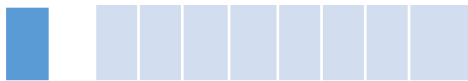
- Xor – exclusive or

1 0 1 1 0 0 1 0

XOR

0 1 0 1 1 0 0 1

=



Carry bit



Overflow

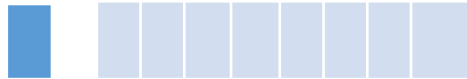
Src 0	Src 1	XOR
0	0	0
0	1	1
1	0	1
1	1	0

\*this operation may or may not modify the carry and overflow bits depending on the particular processor brand (Intel, ARM, IBM...)

# Unary logical operation

- Not

1 0 1 1 0 0 1 0



Carry bit

Overflow

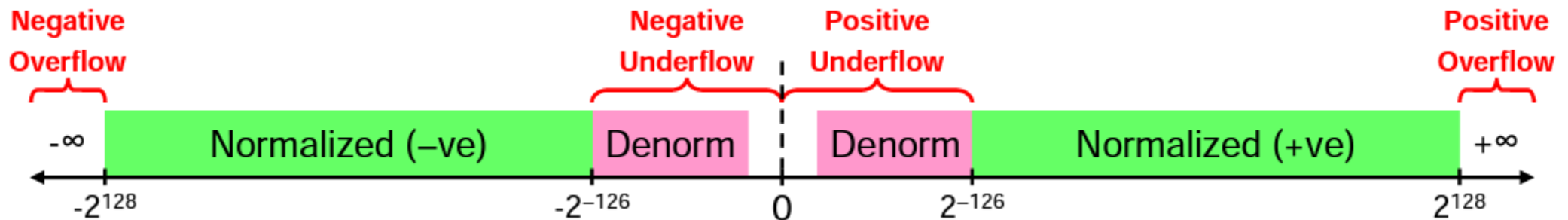
Src	NOT
0	1
1	0

\*this operation may or may not modify the carry and overflow bits depending on the particular processor brand (Intel, ARM, IBM...)

# Floating point

- Underflow vs Overflow
- Normalized vs Denormalized

Single-Precision	Exponent = 8	Fraction = 23	Value
Normalized Number	1 to 254	Anything	$\pm (1.F)_2 \times 2^{E-127}$
Denormalized Number	0	nonzero	$\pm (0.F)_2 \times 2^{-126}$
Zero	0	0	$\pm 0$
Infinity	255	0	$\pm \infty$
NaN	255	nonzero	NaN



# Floating point

---

- $19,59375_{10}$ 
  - 1st Step: determine sign bit
  - 2nd Step: convert to binary
    - $19 \rightarrow$
    - $0,59375 \rightarrow$
  - 3rd Step: normalise the mantissa and determine the unbiased exponent
  - 4th Step: determine the biased exponent
  - 5th Step: remove the leading 1 from the mantissa
- Sol: **0 10000011 001110011100000000000000**

# Floating point

---

•  $0,09375_{10}$

• Sol: **0 01111011 100000000000000000000000**

•  $-123,3_{10}$

• Sol: **1 10000101 11101101001100110011001**

Sol: **1 10000101 11101101001100110011010**

•  $0\ 10000100\ 110101000000000000000000_2$

• Sol: **58,5**

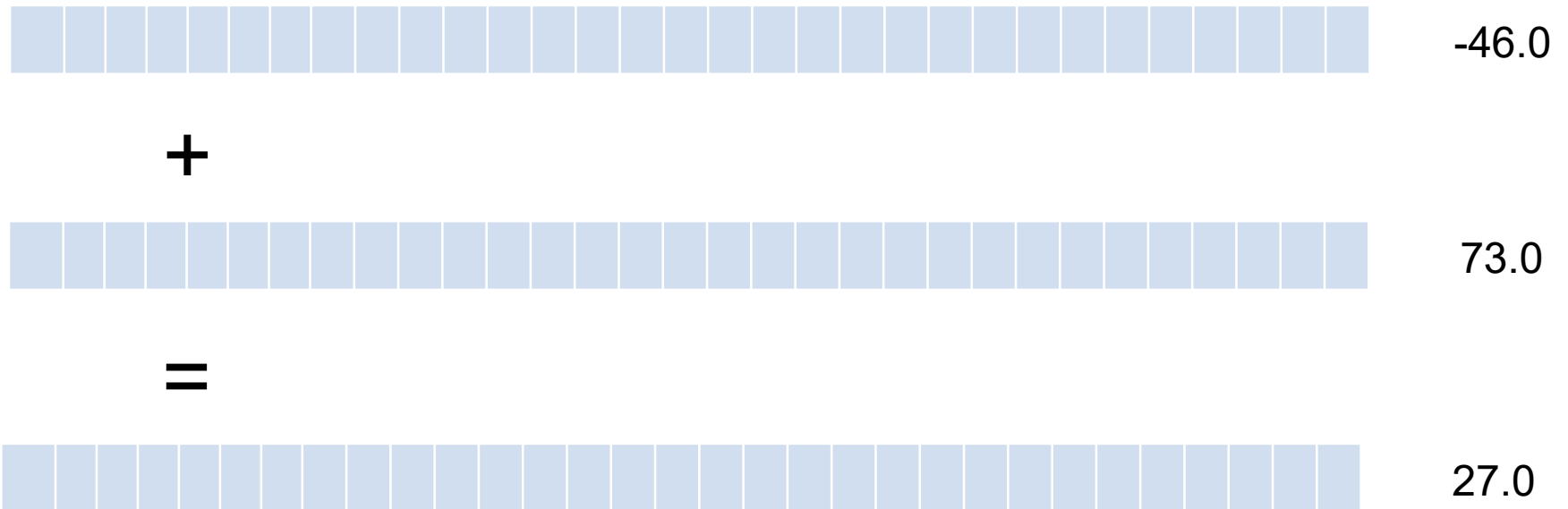
•  $1\ 00001001\ 000110010000000000000000_2$

• Sol: **-3,3031391258106278973921496695945 \*  $10^{-36}$**

# Floating point

---

- Sign + exponent + mantissa



# Binary vs Logical Operations

Expression	Binary value	Hex value
a	0101 0101	
b	0100 0110	
a & b		
a   b		
a ^ b		
~a   ~b		
a & !b		
a && b		
a    b		
!a    !b		
a && ~b		