

# Computers

## Data Representation

### Exercises

*Grau en Ciència i Enginyeria de Dades*

---

**Xavier Martorell, Xavier Verdú**

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2020-2021 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

$$\begin{array}{ccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 2^7 & & 2^5 & 2^4 & & & 2^1 & \end{array} \quad 128+32+16+2 = 178$$

+

$$\begin{array}{ccccccc} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 2^6 & & 2^4 & 2^3 & & & 2^0 & \end{array} \quad 64+16+8+1 = 89$$

=

$$\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 2^8 & & & & & 2^3 & 2^1 & 2^0 & \end{array} \quad 256 + 8 + 2 + 1 = 267$$

Correct if extended to 9 bits!!!

$$\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \text{Carry bit} & & & & & 2^3 & 2^1 & 2^0 & \end{array} \quad 8 + 2 + 1 = 11$$

Otherwise -> Incorrect representation and Carry bit is 1!!

# Binary arithmetic

---

- Substraction

$$\begin{array}{r} 10110010 \\ \underline{01011001} \\ 01011001 \\ \text{Carry bit } 0 \end{array}$$

$2^7 \quad 2^5 \quad 2^4 \quad 2^1 \quad 128+32+16+2 = 178$

$2^6 \quad 2^4 \quad 2^3 \quad 2^0 \quad 64+16+8+1 = 89$

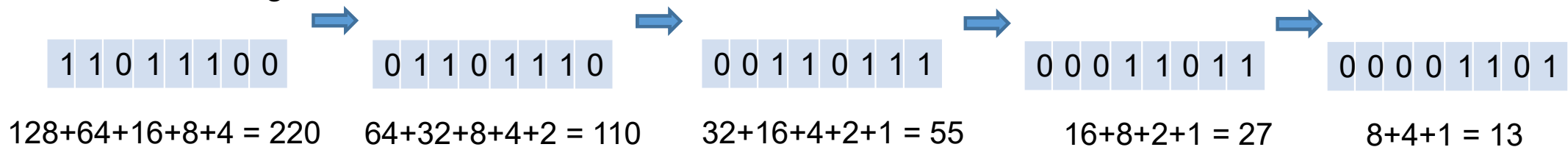
$2^6 \quad 2^4 \quad 2^3 \quad 2^0 \quad 64 + 16 + 8 + 1 = \mathbf{89}$

- **Exercise:** do you see a property shared between the two numbers
  - 10110010 and...
  - 01011001 that explains why the result is half of the first number?

# Binary arithmetic

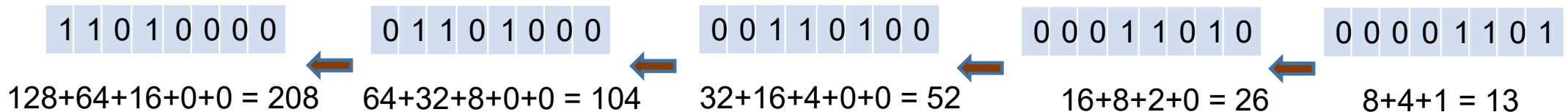
- Division /2 - shift right

- 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

$$\begin{array}{r}
 \begin{array}{cccccc}
 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
 -2^7 & 2^6 & & 2^4 & & & 2^1 & \\
 \hline
 & + & & & & & & \\
 \hline
 \begin{array}{cccccc}
 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
 & & 2^5 & 2^4 & & 2^2 & 2^1 & \\
 \hline
 & = & & & & & & \\
 \hline
 \begin{array}{cccccc}
 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 \text{Carry bit} & & & & & 2^3 & & & 8 \\
 \leftarrow & \leftarrow & & & & & & & \\
 1 & 1 & & & & & & & 
 \end{array}
 \end{array}
 \end{array}
 \quad -128+64+16+2 = -46$$

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

- And the result is correctly represented in 8 bits -> **no overflow!!**

# Binary arithmetic

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

$$\begin{array}{r}
 \begin{array}{ccccccc}
 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
 & 2^6 & & 2^4 & & & 2^1 & \\
 \hline
 & + & & & & & & \\
 \hline
 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
 & 2^5 & 2^4 & & 2^2 & 2^1 & & \\
 \hline
 = & & & & & & & \\
 \hline
 \begin{array}{ccccccc}
 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 & & & & & 2^3 & & & \\
 \hline
 \text{Carry bit} & \leftarrow & & & & & & & \\
 0 & & 1 & & & & & & 
 \end{array}
 \end{array}
 \end{array}
 \begin{array}{l}
 64+16+2 = 82 \\
 \\
 32+16+4+2 = 54 \\
 \\
 -128+8 = -120
 \end{array}$$

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

- And the result is **incorrectly** represented in 8 bits -> **overflow!!**

# Binary logical operations

- And

$1\ 0\ 1\ 1\ 0\ 0\ 1\ 0$   
 $2^7\ 2^5\ 2^4\ 2^1$       $128+32+16+2 = 178$

AND

$0\ 1\ 0\ 1\ 1\ 0\ 0\ 1$   
 $2^6\ 2^4\ 2^3\ 2^0$       $64+16+8+1 = 89$

=

0  $0\ 0\ 0\ 1\ 0\ 0\ 0\ 0$   
 $2^4$      16

Carry bit

0

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  
2<sup>7</sup> 2<sup>5</sup>2<sup>4</sup> 2<sup>1</sup>

128+32+16+2 = 178

OR

0 1 0 1 1 0 0 1  
2<sup>6</sup> 2<sup>4</sup>2<sup>3</sup> 2<sup>0</sup>

64+16+8+1 = 89

=

0 1 1 1 1 0 1 1

128+64+32+16+8+2+1 = 251

Carry bit

0

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

$1\ 0\ 1\ 1\ 0\ 0\ 1\ 0$   
 $2^7\ 2^5\ 2^4\ 2^1$       $128+32+16+2 = 178$

XOR

$0\ 1\ 0\ 1\ 1\ 0\ 0\ 1$   
 $2^6\ 2^4\ 2^3\ 2^0$       $64+16+8+1 = 89$

=

$0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1$   
 $128+64+32+8+2+1 = 235$

Carry bit

0

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  
2<sup>7</sup> 2<sup>5</sup>2<sup>4</sup> 2<sup>1</sup>

$$128+32+16+2 = 178$$



0 0 1 0 0 1 1 0 1

$$64+8+4+1 = 77$$

Carry bit

0

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

---

- Sign + exponent + mantissa

		46.0	$1.011100 \cdot 10^{(10000100-01111111)}$	
1	10000100	1.	0111000000000000000000	-46.0
		+		
		73.0	$1.001001 \cdot 10^{(10000101-01111111)}$	
0	10000101	1.	0010010000000000000000	73.0
		=		
		27.0	$1.101100 \cdot 10^{(10000011-01111111)}$	
0	10000011	1.	1011000000000000000000	27.0

# Floating point

---

- Sign + exponent + mantissa

$$\begin{aligned} -46.0 & \quad 1.011100 * 10^{(10000100-01111111)} \\ & \quad 1.437500 * \mathbf{2^5} = 1.4375 * 32 = 46.0 \end{aligned}$$

$$\begin{aligned} 73.0 & \quad 1.001001 * 10^{(10000101-01111111)} \\ & \quad 1.140625 * \mathbf{2^6} = 1.140625 * 64 = 73.0 \end{aligned}$$

$$\begin{aligned} 27.0 & \quad 1.101100 * 10^{(10000011-01111111)} \\ & \quad 1.6875 * \mathbf{2^4} = 1.687500 * 16 = 27.0 \end{aligned}$$

Using \$ bc

# Floating point

---

- Sign + exponent + mantissa

-46.0    **0.1011100\*2<sup>6</sup>**

73.0    **1.0010010\*2<sup>6</sup>**

27.0    **0.0110110\*2<sup>6</sup>**

27.0    **1.1011000\*2<sup>4</sup>**



**subtract**

**normalize**

# Character encodings

---

- Additionally to the characters
  - 64 - @
  - 65 – A      97 – a
  - 66 – B      98 – b
  - ...
- Character properties
  - isalpha(c), isdigit, isalnum, islower, isupper, isspace...    ASCII
  - u\_isdefined(uc), u\_isdigit, u\_isalpha, u\_islower...          Unicode