Computers
Data Representation Exercises

Grau en Ciència i Enginyeria de Dades

Xavier Martorell, Xavier Verdú

Facultat d'Informàtica de Barcelona (FIB)

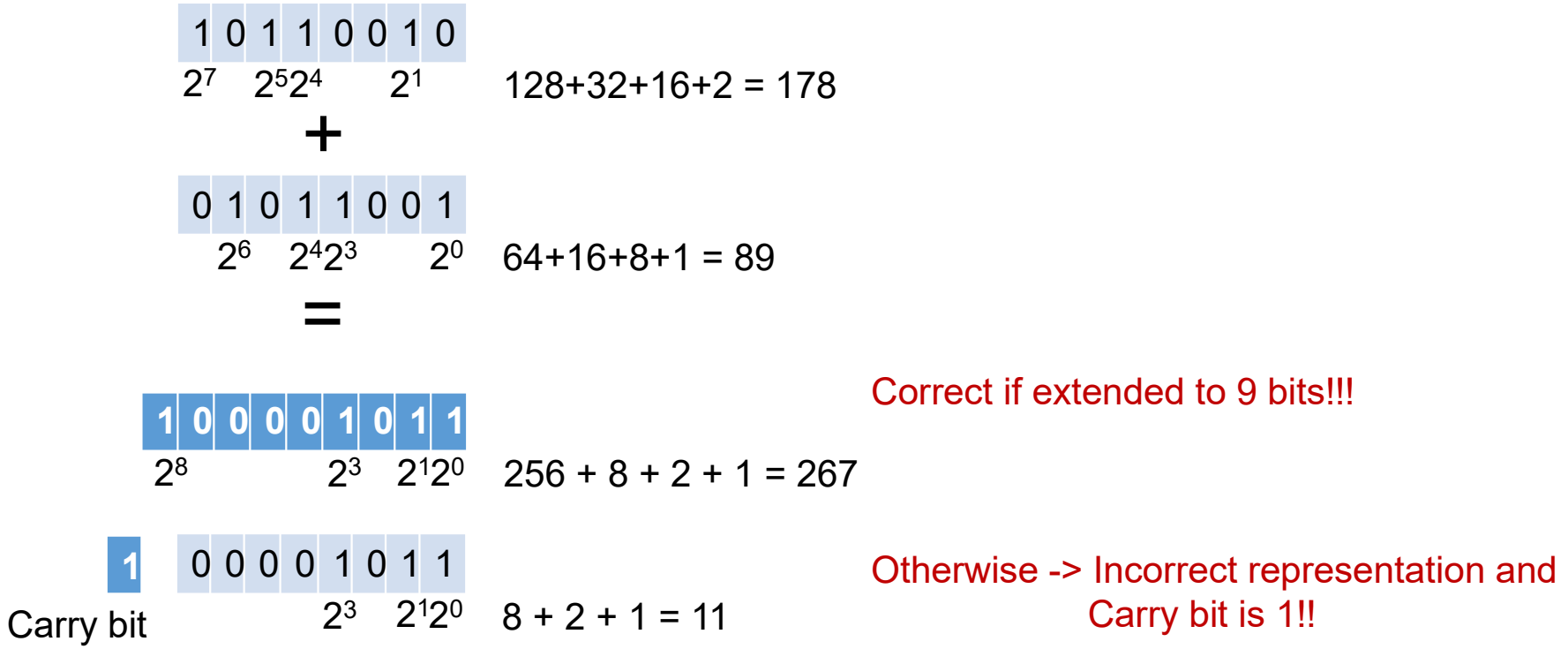
Universitat Politècnica de Catalunya (UPC)

2020-2021 Q2

Creative Commons License

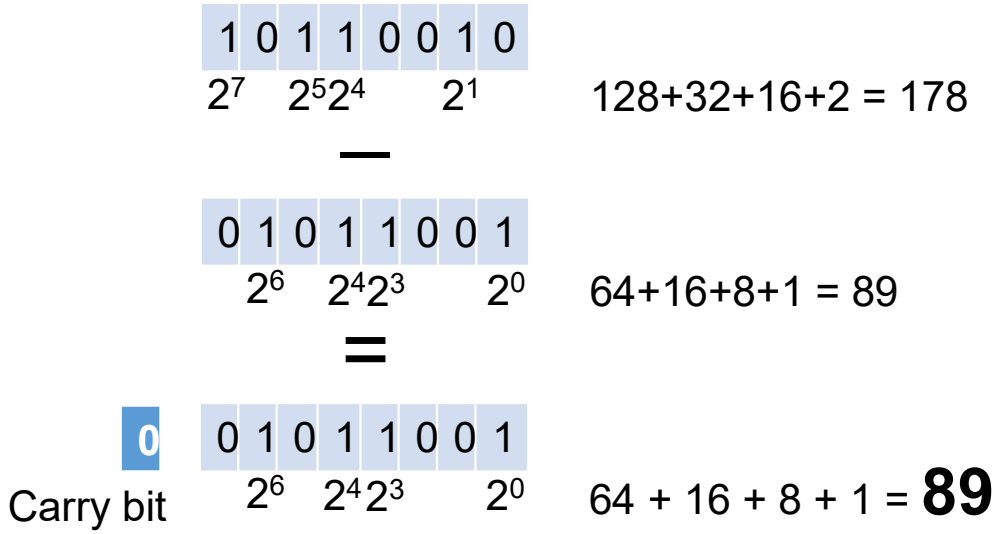
Binary arithmetic

- Binary addition



Binary arithmetic

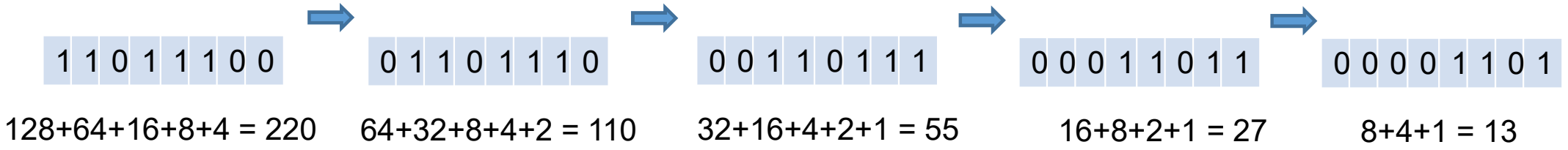
- Substraction



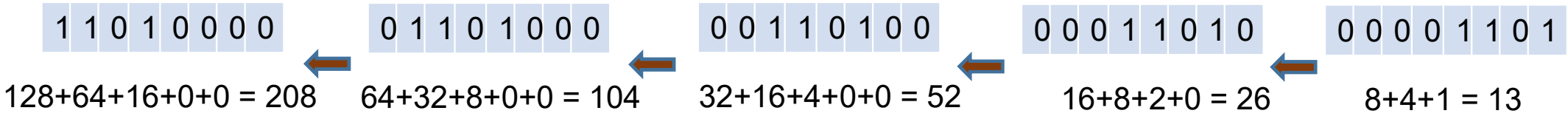
- Exercise: do you see a property shared between the two numbers
  - 10110010 and...
  - 01011001 that explains why the result is half of the first number?

Binary arithmetic

- Division /2 - shift right
  - Right-most bit is lost

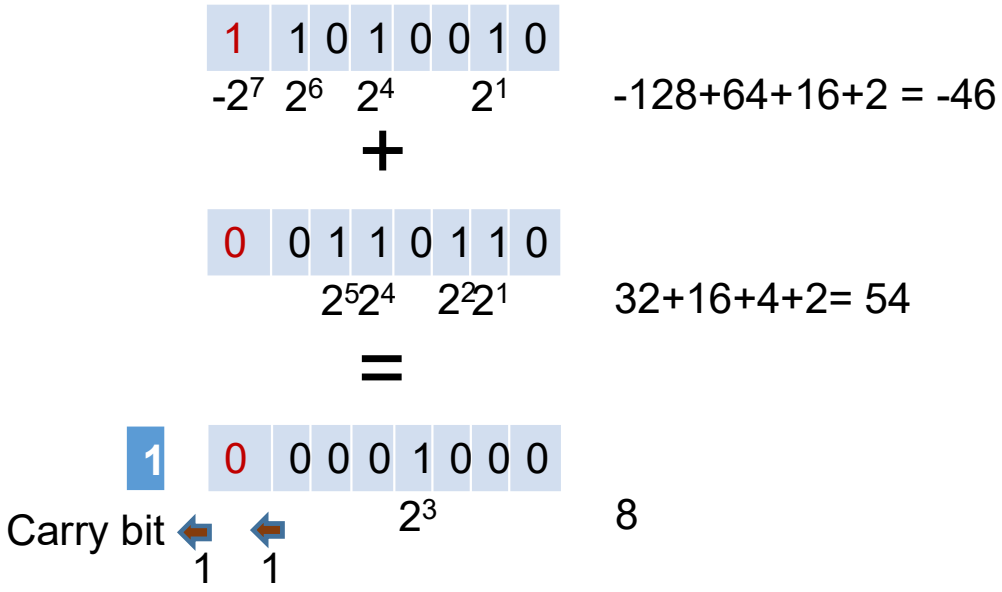


- Multiplication*2 - shift left
  - Left-most bit gets into the carry



Binary arithmetic

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

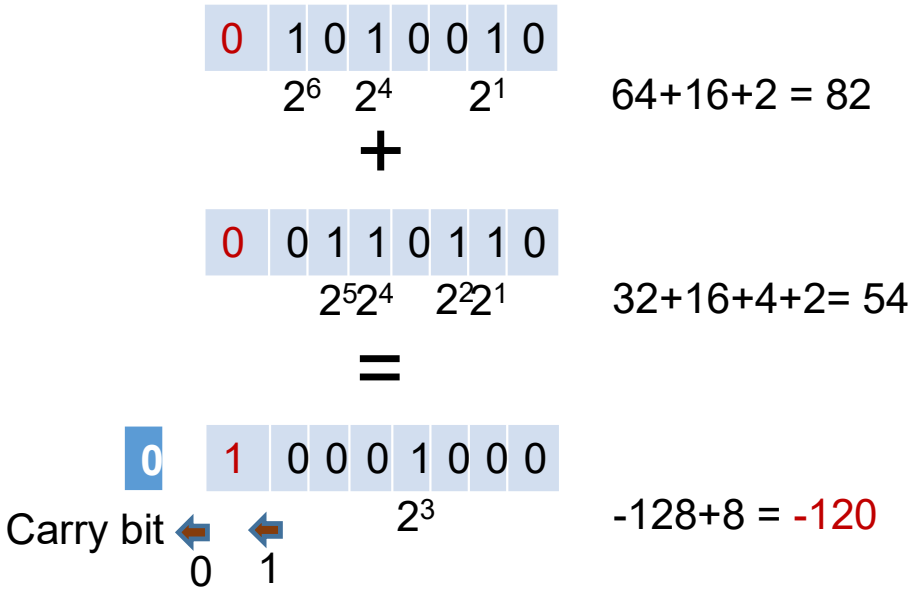


Into sign	Into carry	Overflow
0	0	0
0	1	
1	0	
1	1	0

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

Binary arithmetic

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



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	1	0	0	0	0

2^4 16

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

0 Carry bit

0 Overflow

*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
2^7		2^5	2^4			2^1	

128+32+16+2 = 178

OR

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	1	1	0	1	1

Carry bit

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

0

Overflow

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

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

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

Carry bit

128+64+32+8+2+1 = 235

0

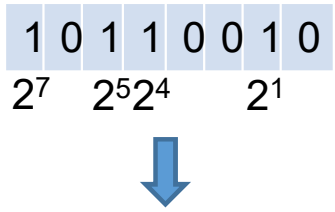
Overflow

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

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

Unary logical operation

- Not



128+32+16+2 = 178

Src	NOT
0	1
1	0

64+8+4+1 = 77

Carry bit

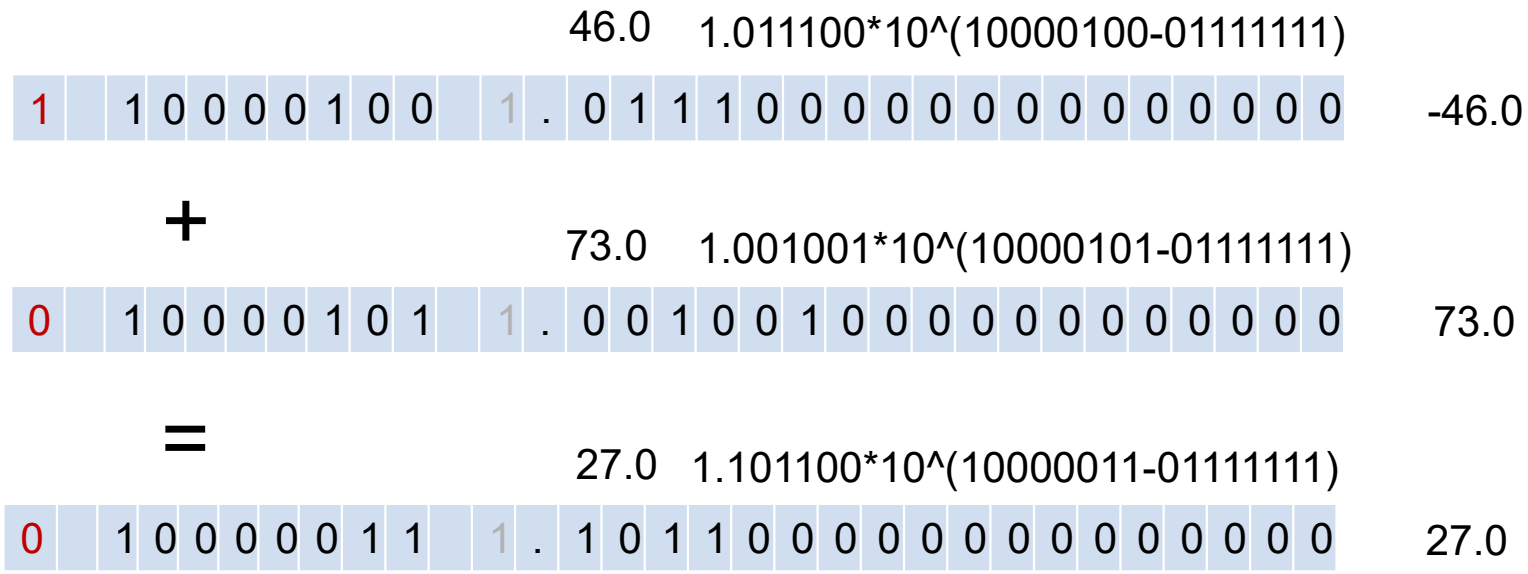
0

Overflow

*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



Floating point

- Sign + exponent + mantissa

-46.0 1.011100*10^(10000100-01111111)

1.437500 * **2^5** = 1.4375 * 32 = 46.0

73.0 1.001001*10^(10000101-01111111)

1.140625 * **2^6** = 1.140625 * 64 = 73.0

27.0 1.101100*10^(10000011-01111111)

1.6875 * **2^4** = 1.687500 * 16 = 27.0

Using $ bc

Floating point

- Sign + exponent + mantissa

-46.0 **0.**1011100***2^6**

73.0 1.0010010***2^6**

27.0 0.0110110*2^6

27.0 1.1011000***2^4**

substract

normalize

Using $ bc

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