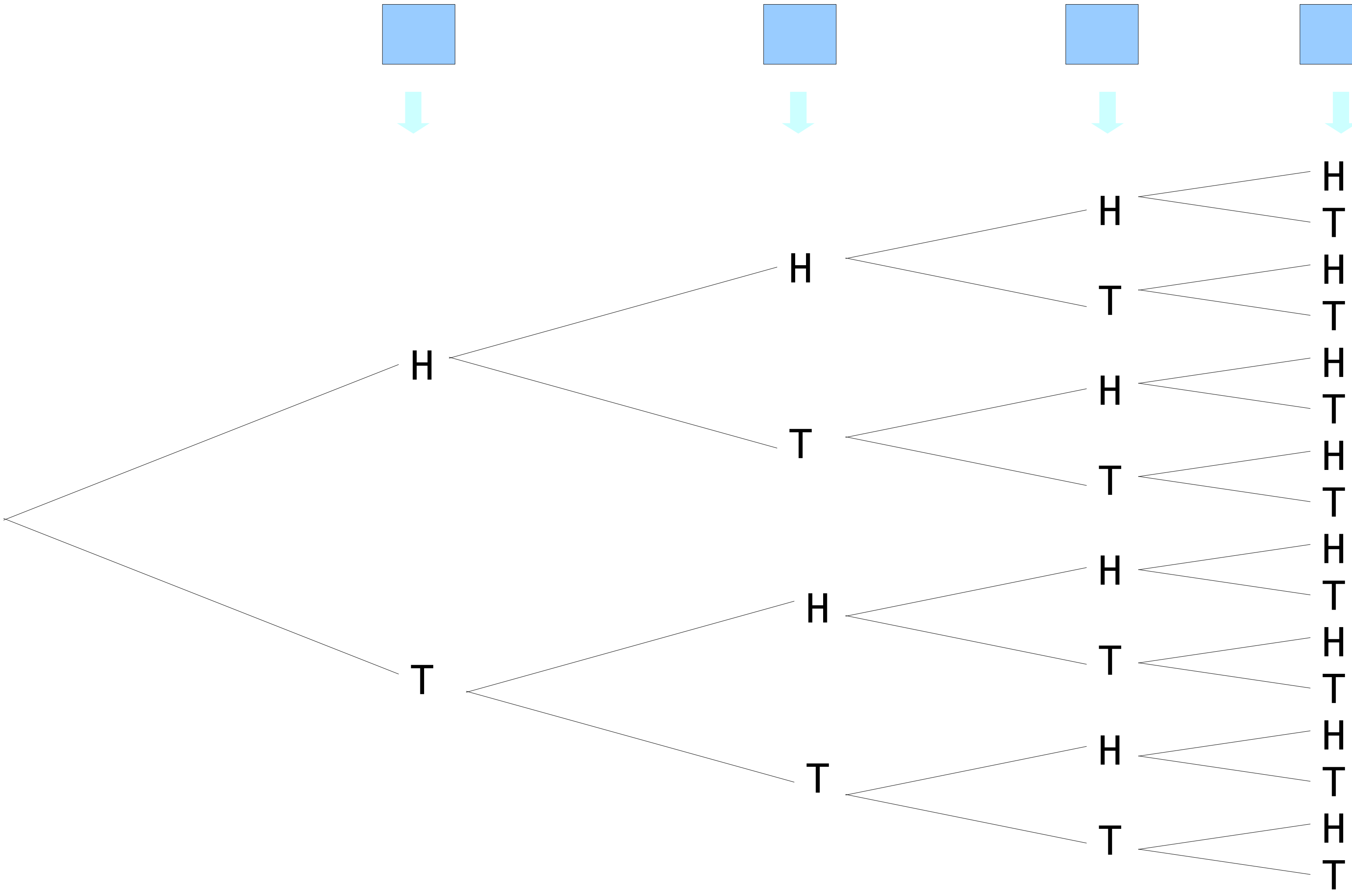
Number System (1A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

T0is document was produced by using OpenOffice/LibreOffice.

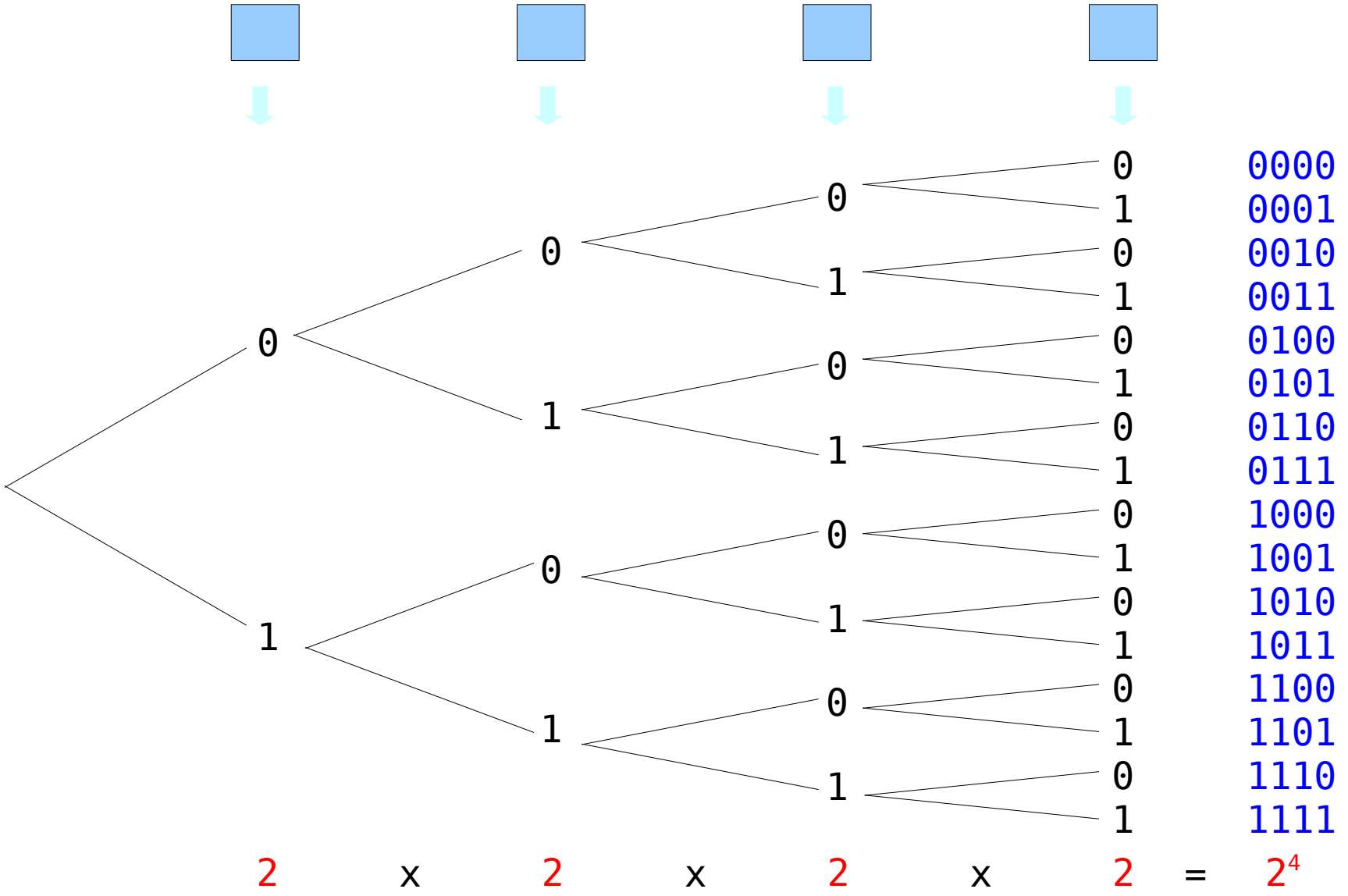
Young Won Lim
7/7/10

Coin Tossing



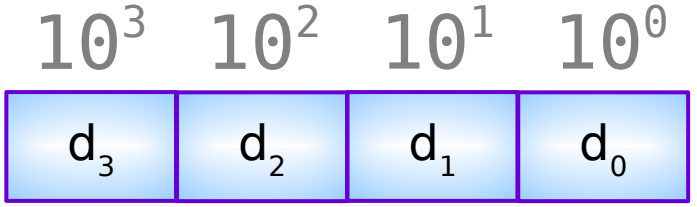
$2 \times 2 \times 2 \times 2 = 2^4$

Coin Tossing and Binary Number

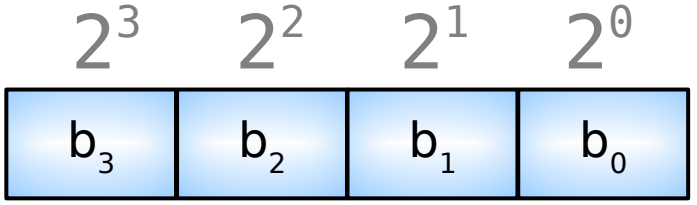


Radix Number Systems

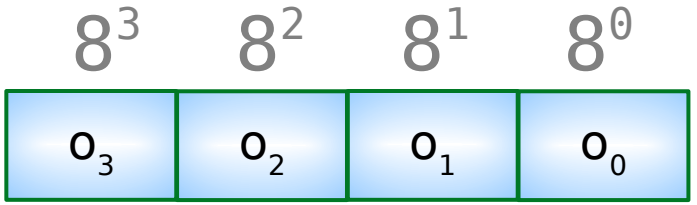
Decimal



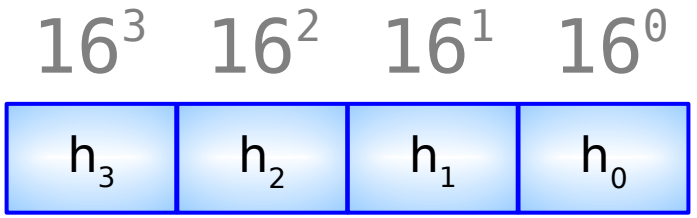
Binary



Octal



Hexadecimal



Conversion to decimal numbers

Decimal

$$d_3 \cdot 10^3 + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

Binary

$$b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

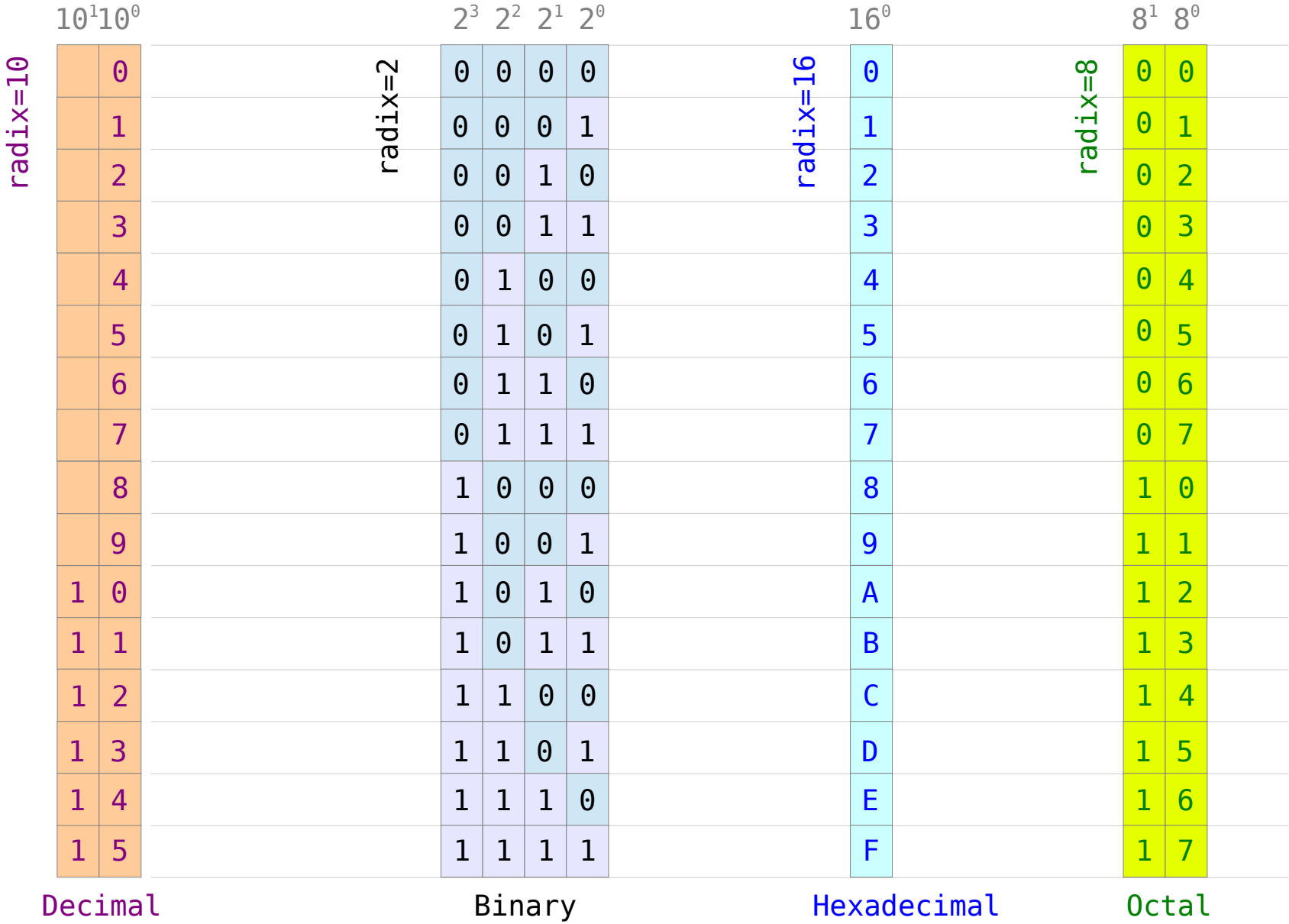
Octal

$$o_3 \cdot 8^3 + o_2 \cdot 8^2 + o_1 \cdot 8^1 + o_0 \cdot 8^0$$

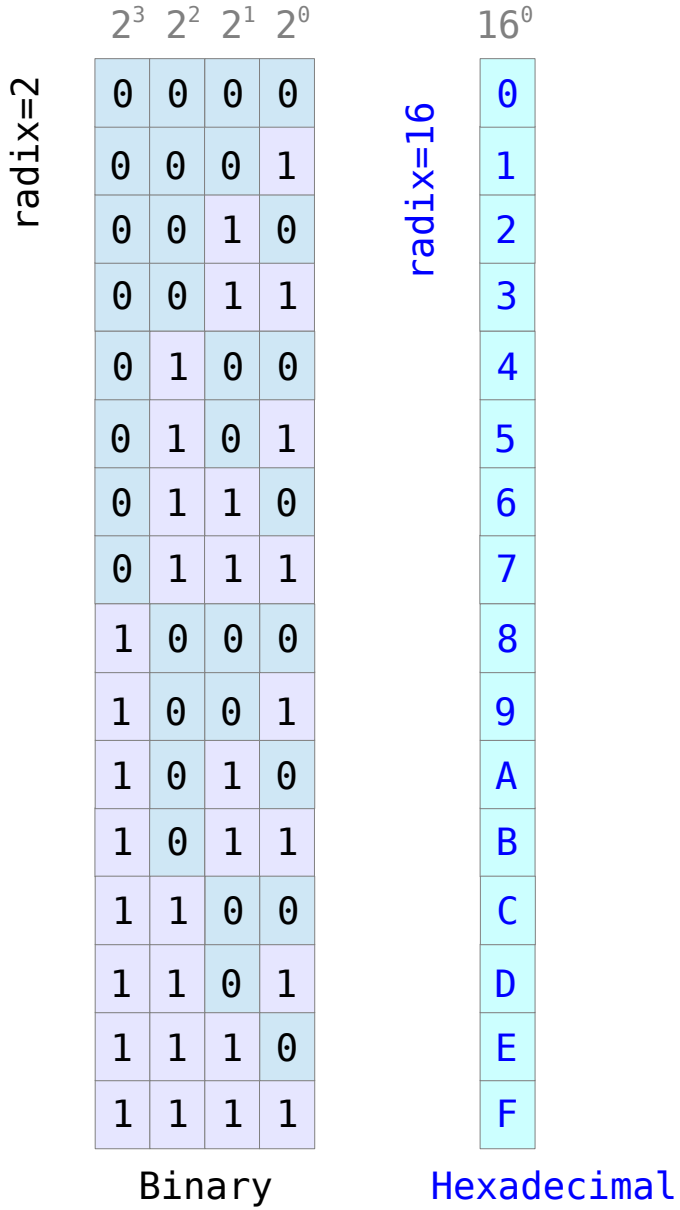
Hexadecimal

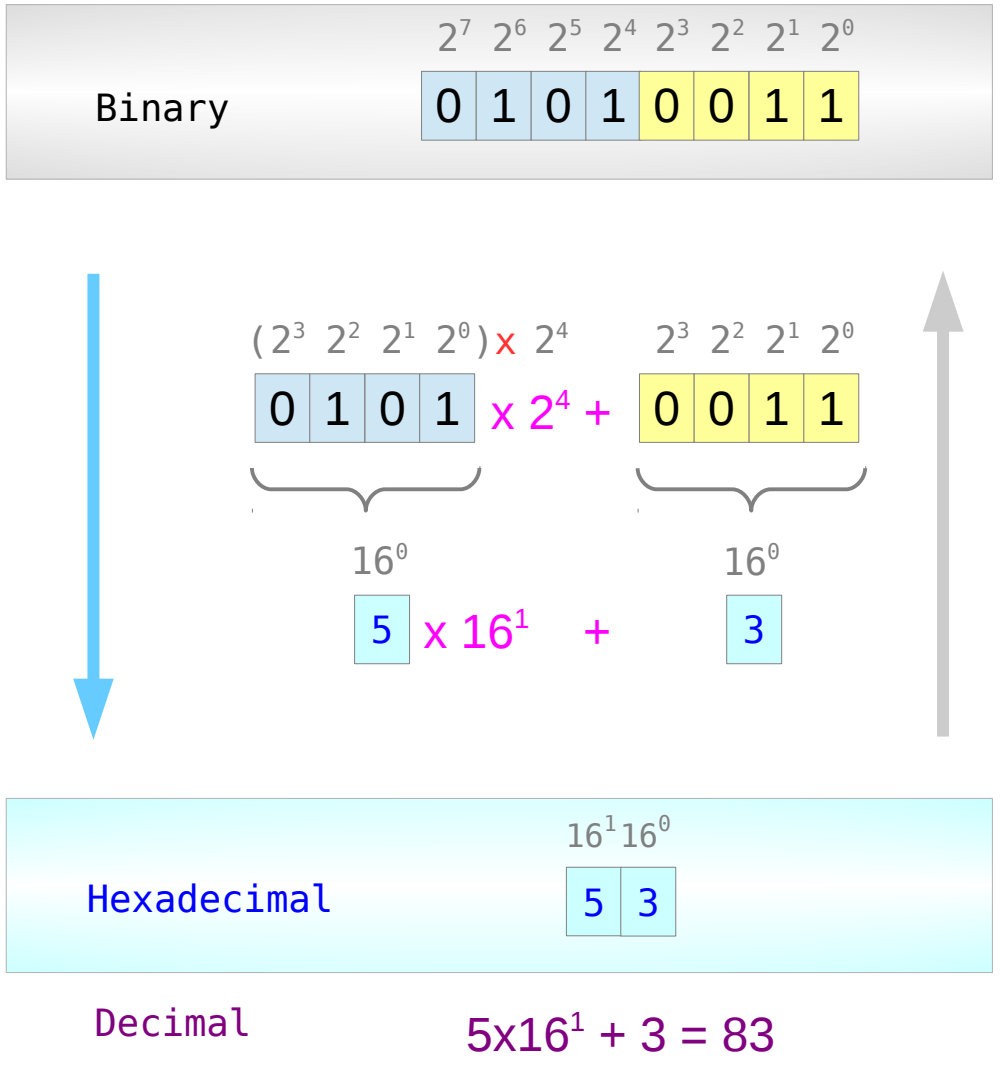
$$h_3 \cdot 16^3 + h_2 \cdot 16^2 + h_1 \cdot 16^1 + h_0 \cdot 16^0$$

Number Systems [0..15]

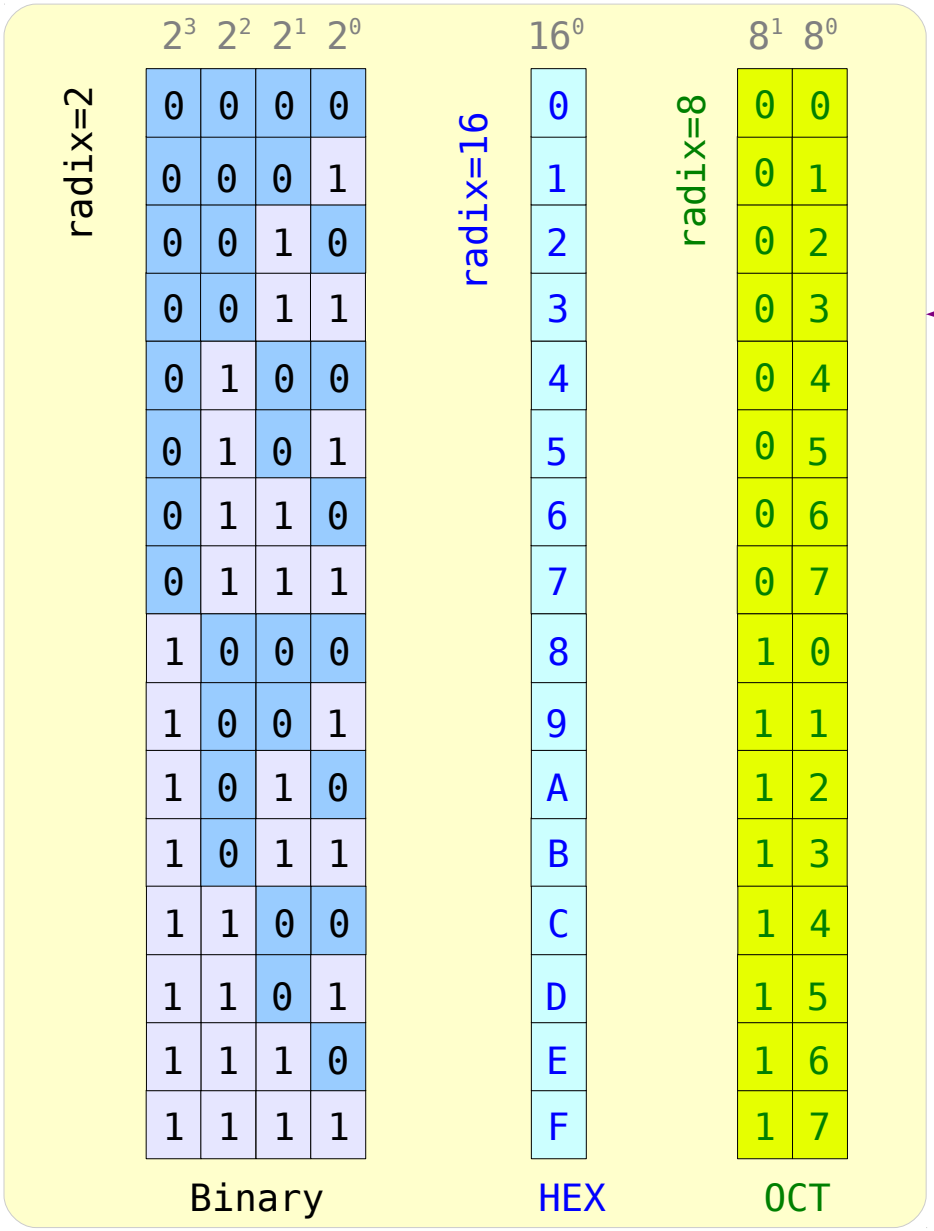


Binary and Hexadecimal Numbers





Bit Patterns

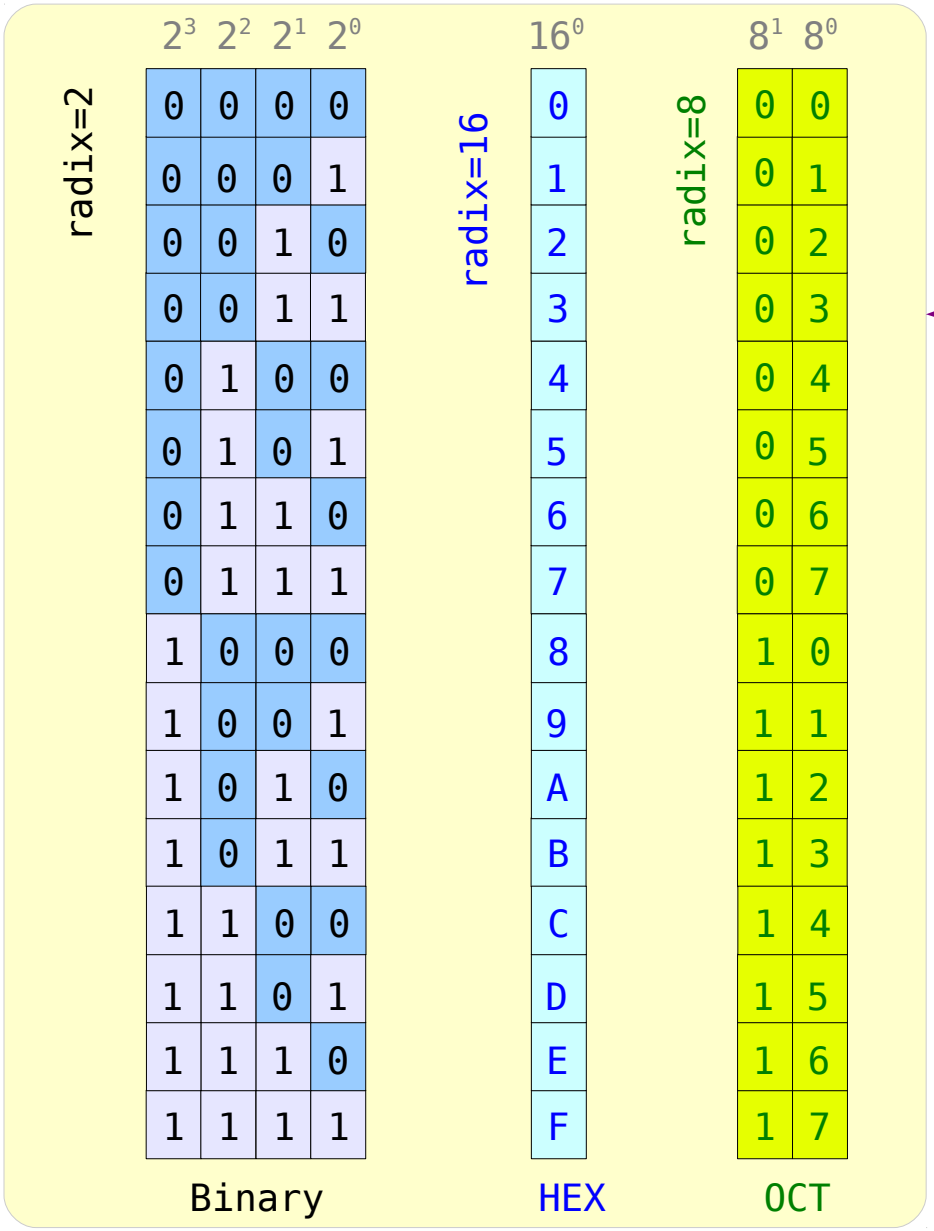


Bit patterns stored in a computer memory system

- Unsigned
- Signed

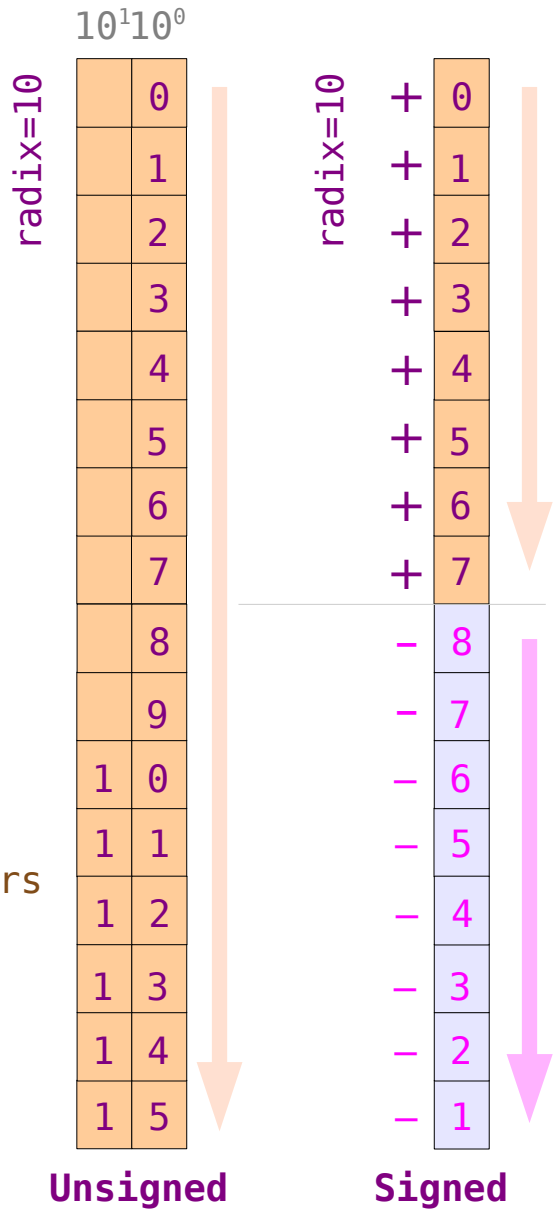
Each bit pattern can represent either unsigned or signed numbers

Signed and Unsigned Numbers

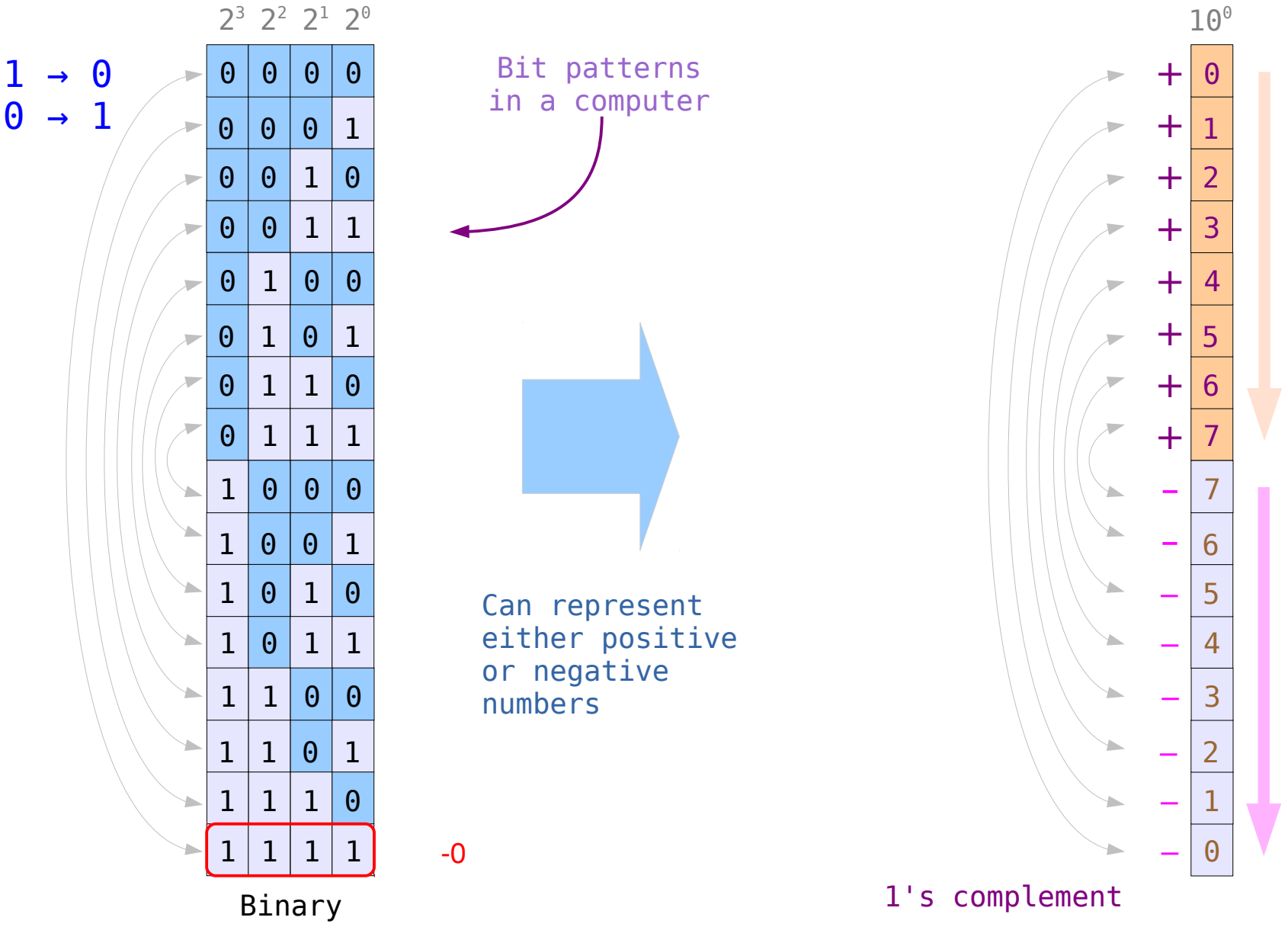


Bit patterns
in a computer

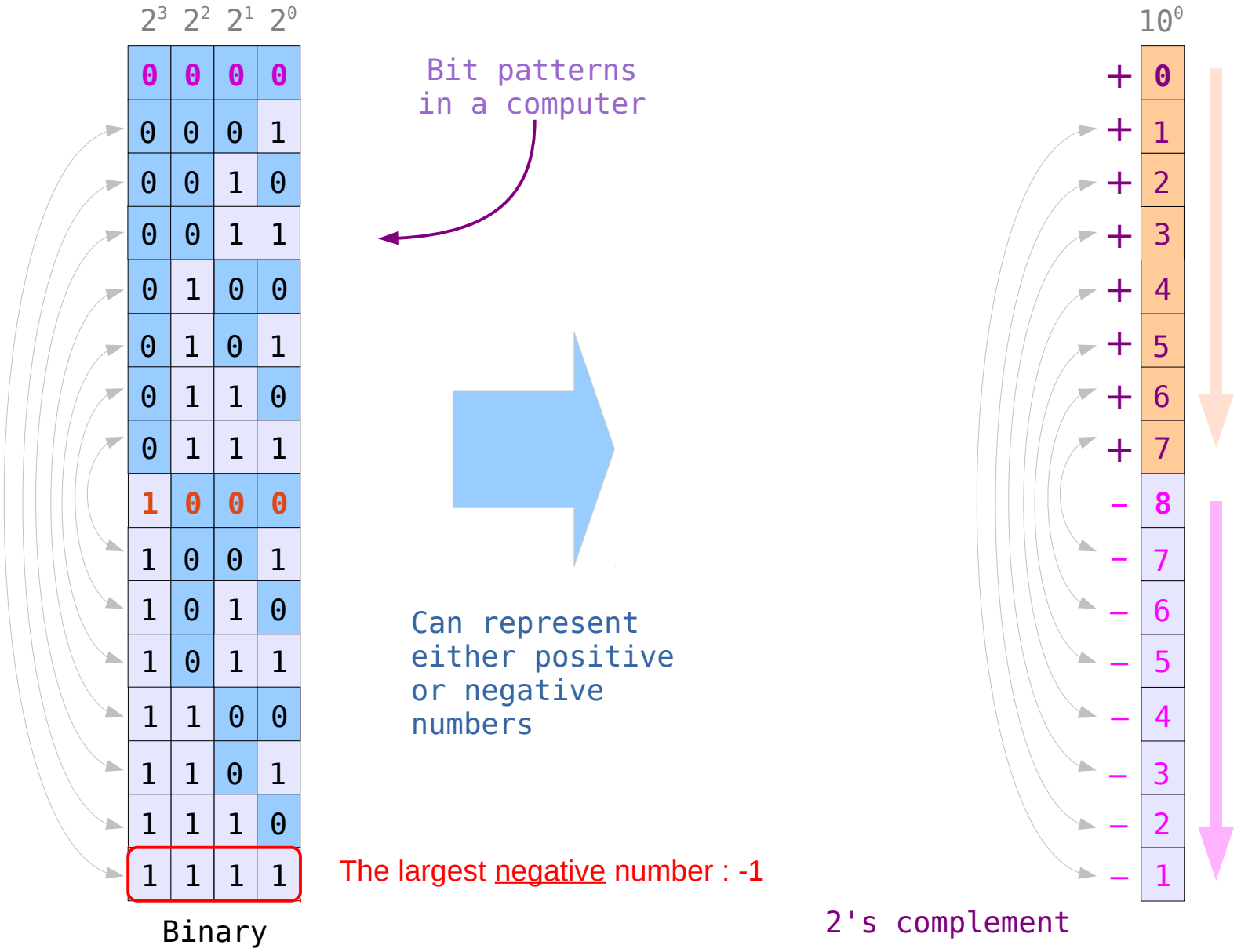
Can represent
either unsigned
or signed numbers



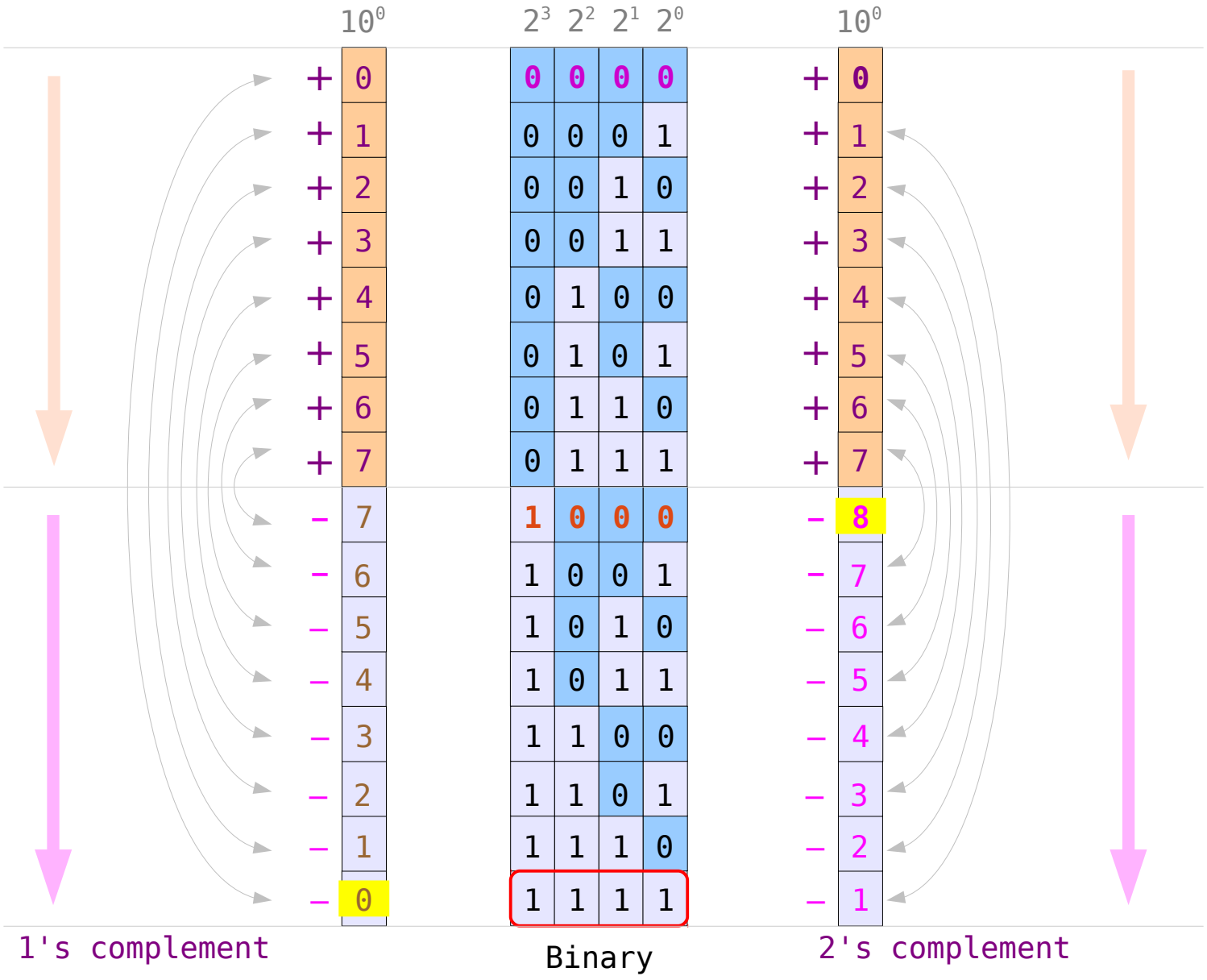
1's complement **signed** numbers



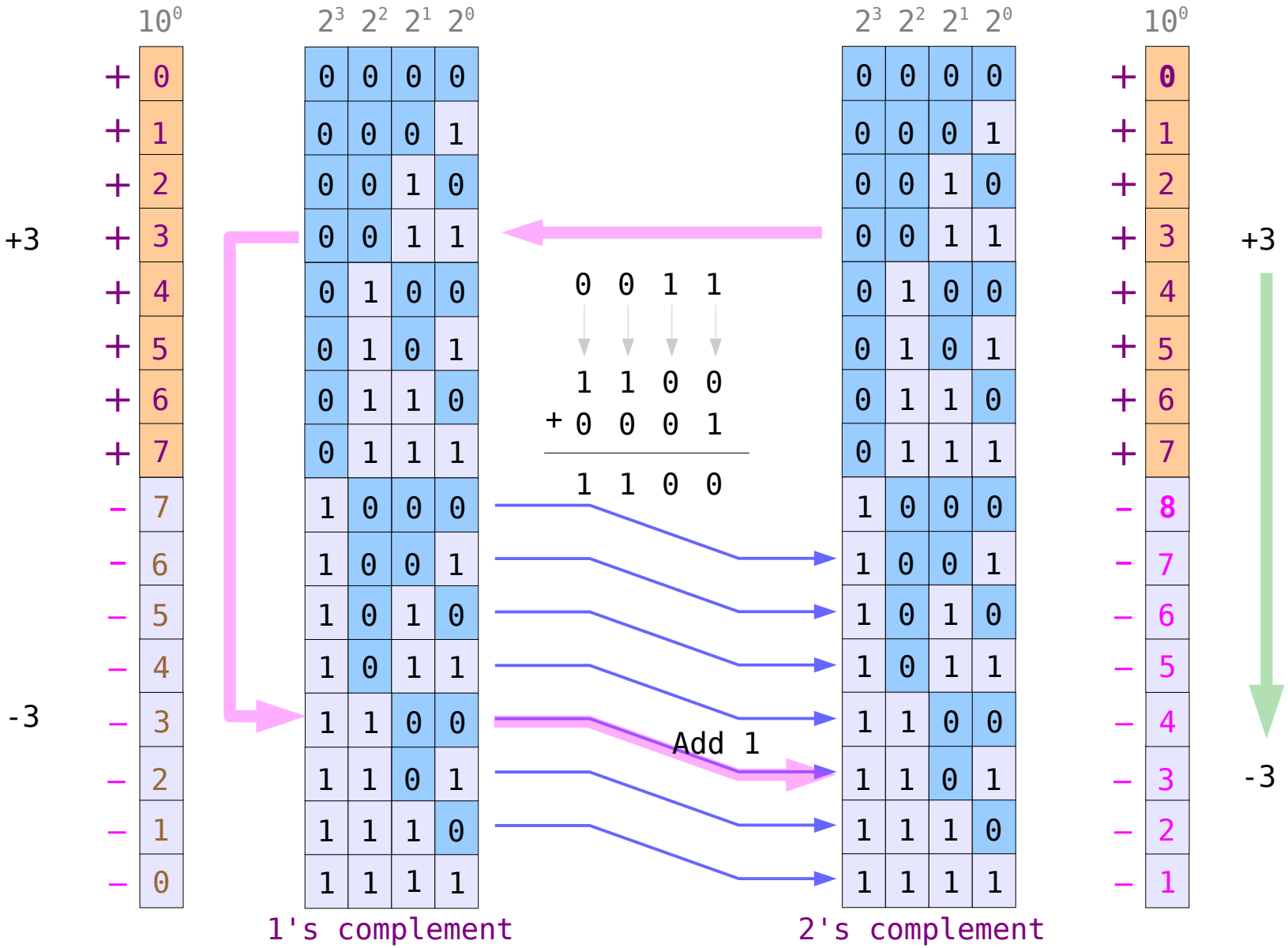
2's complement **signed** numbers



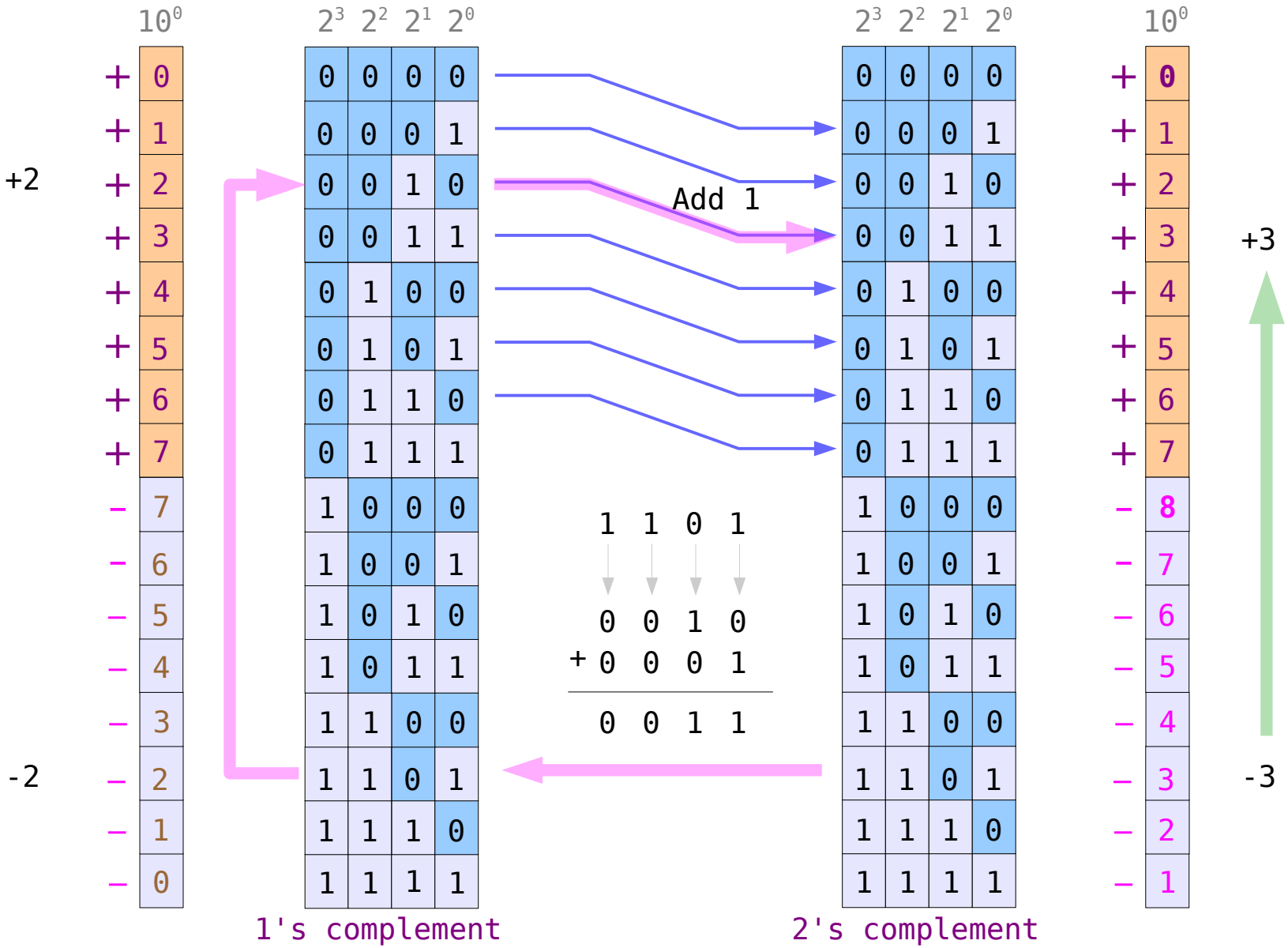
1’s and 2's complement **signed** numbers



Compute 2's Complement (**+3 → -3**)

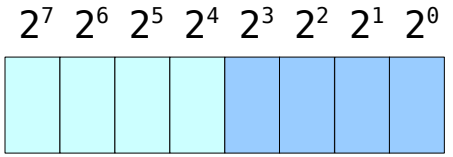


Compute 2's Complement (**-3 → +3**)

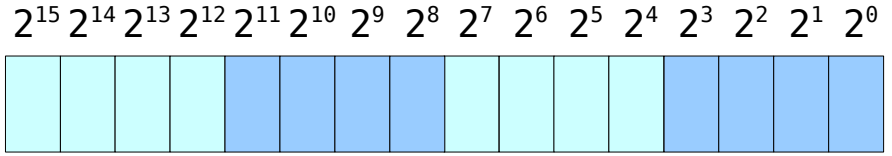


Types of Integer Numbers

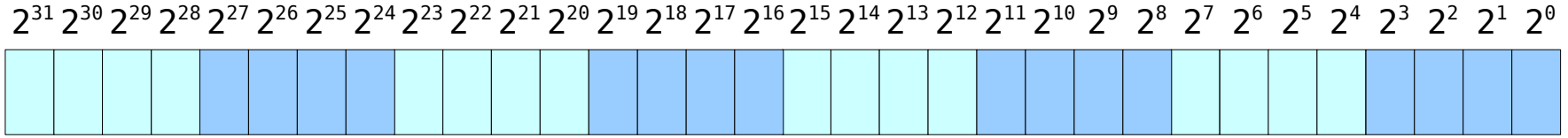
1 byte : **char** 2^8



2 bytes: **short** 2^{16}



4 bytes: **int** 2^{32}



Unsigned Integer Ranges

1 Byte

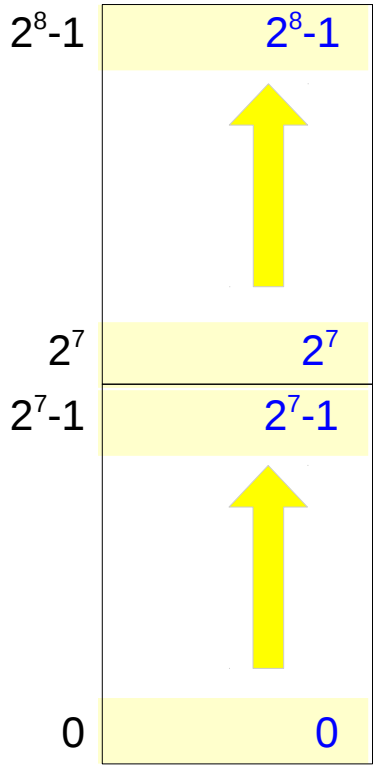
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

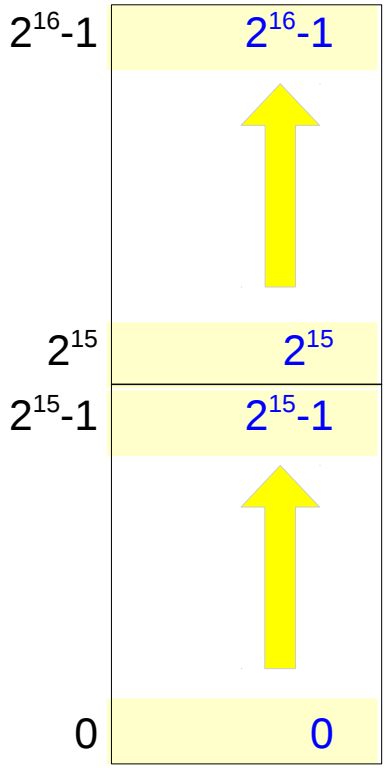
2 Bytes

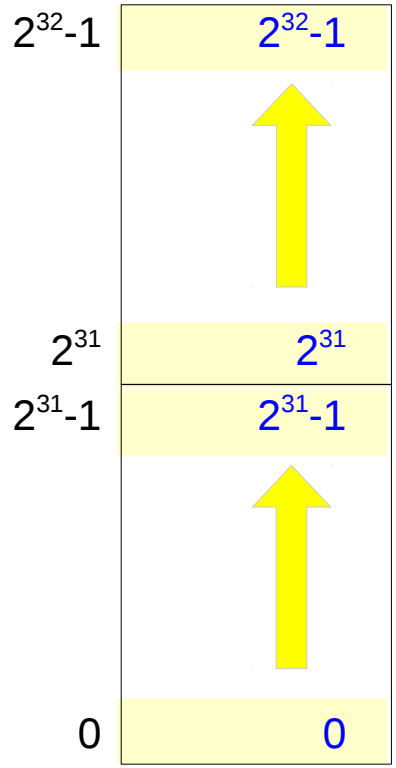
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

4 Bytes

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0







Signed Integer Ranges

1 Byte

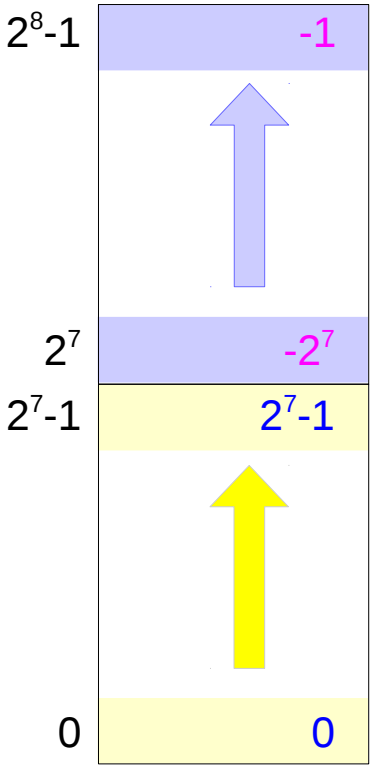
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

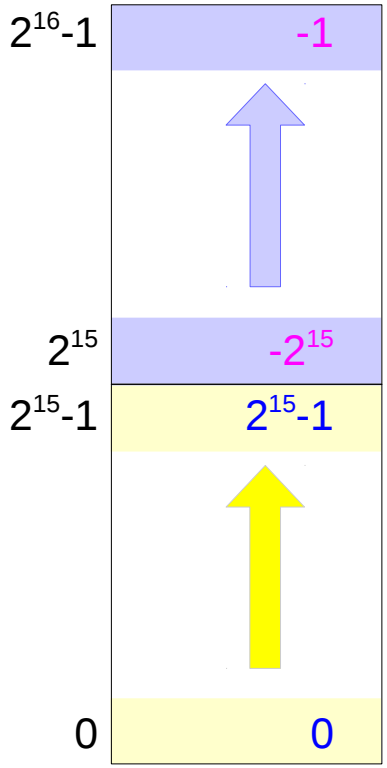
2 Bytes

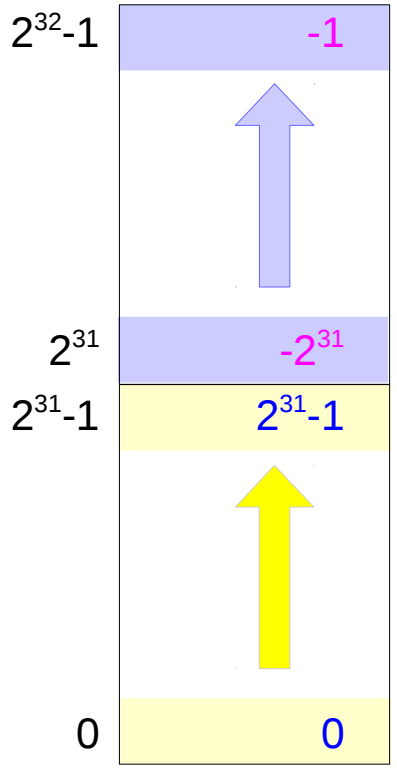
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

4 Bytes

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

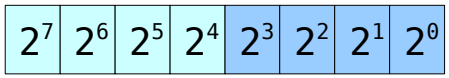




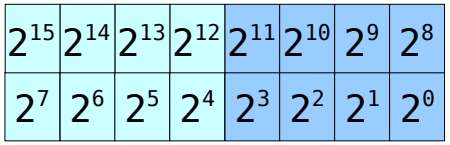


Integer Ranges

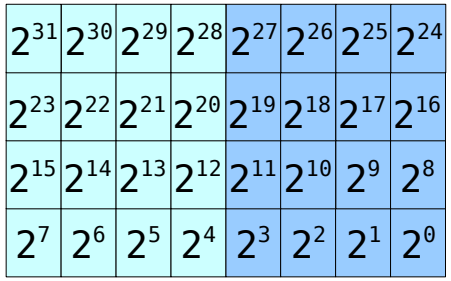
1 Byte



2 Bytes



4 Bytes



unsigned char

$$[0,\ +(2^8-1)]$$

unsigned short

$$[0,\ +(2^{16}-1)]$$

unsigned int

$$[0,\ +(2^{32}-1)]$$

(signed) char

$$[0,\ +(2^7-1)]$$

$$[-2^7,\ -1]$$

(signed) short

$$[0,\ +(2^{15}-1)]$$

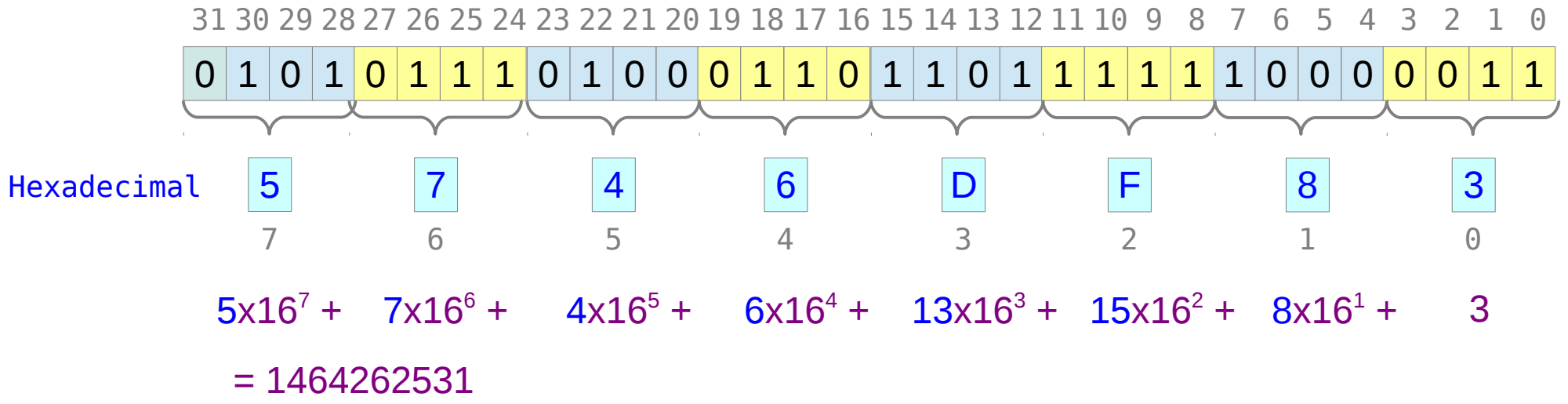
$$[-2^{15},\ -1]$$

(signed) int

$$[0,\ +(2^{31}-1)]$$

$$[-2^{31},\ -1]$$

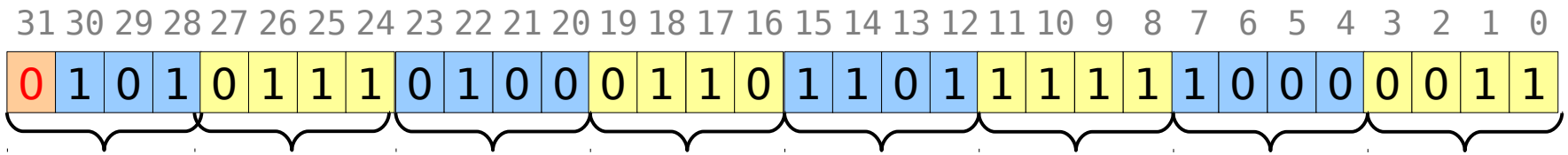
4 Byte **Unsigned** Integer Example



$5x16^7 + 7x16^6 + 4x16^5 + 6x16^4 + 13x16^3 + 15x16^2 + 8x16^1 + 3$

$= 1464262531$

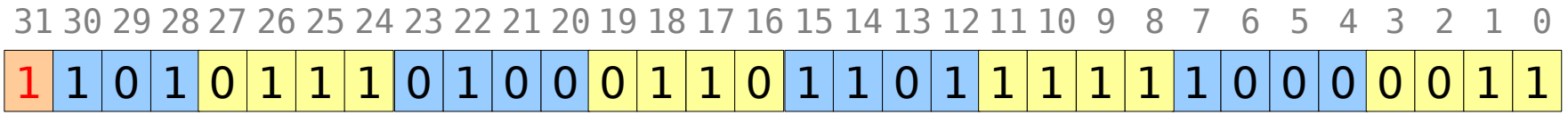
4 Byte **Signed** Integer – positive number



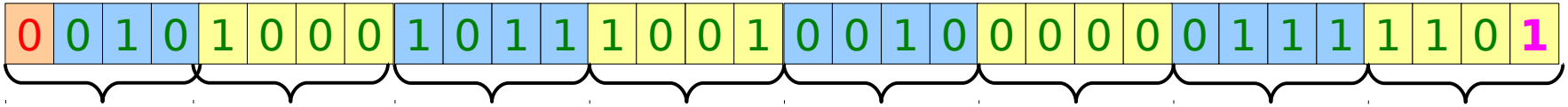
$5x16^7 + 7x16^6 + 4x16^5 + 6x16^4 + 13x16^3 + 15x16^2 + 8x16^1 + 3$

$= + 1464262531$

4 Byte **Signed** Integer – negative number



2's complement



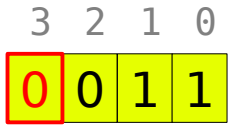
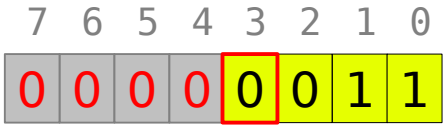
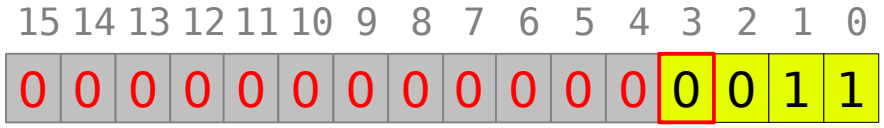
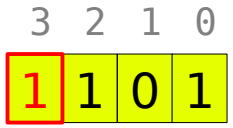
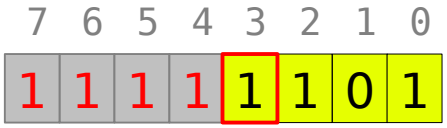
– 2 8 B 9 2 0 7 D

7 6 5 4 3 2 1 0

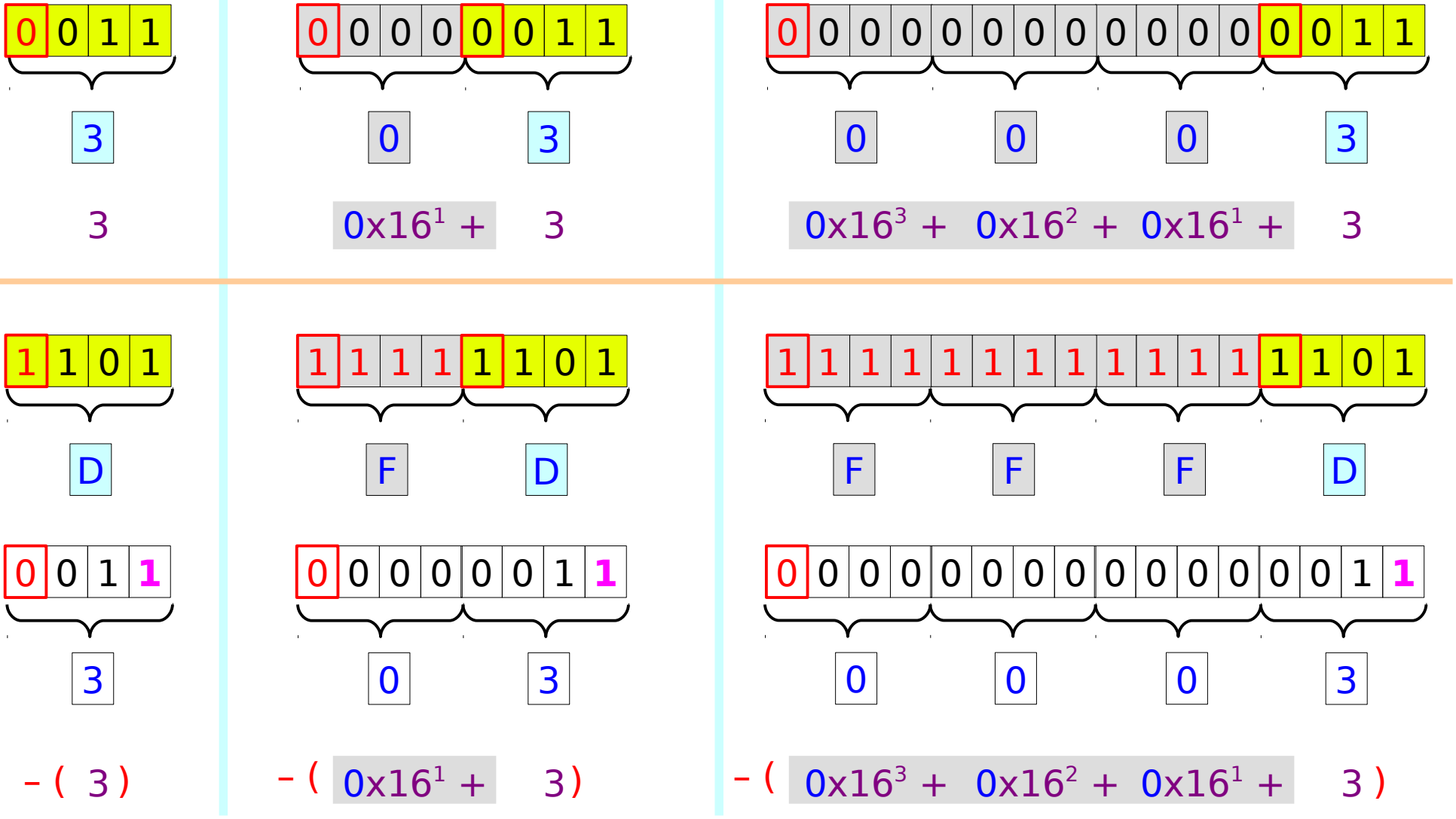
$-(2x16^7 + 8x16^6 + 11x16^5 + 9x16^4 + 2x16^3 + 0x16^2 + 7x16^1 + 13)$

$= -683221117$

The same number with different bit widths

4-bit Signed Number	8-bit Signed Number	16-bit Signed Number
(+3)	(+3)	(+3)
		
(–3)	(–3)	(–3)
		

Sign Extension

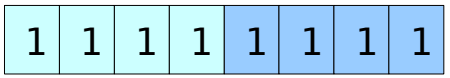


Sign extension in binary and hexadecimal numbers

				16^0					2^3	2^2	2^1	2^0
...	0	0	0	0	...	0	0	0	0	0	0	0
...	0	0	0	1	...	0	0	0	0	0	0	1
...	0	0	0	2	...	0	0	0	0	0	1	0
...	0	0	0	3	...	0	0	0	0	0	1	1
...	0	0	0	4	...	0	0	0	0	1	0	0
...	0	0	0	5	...	0	0	0	0	1	0	1
...	0	0	0	6	...	0	0	0	0	1	1	0
...	0	0	0	7	...	0	0	0	0	1	1	1
...	F	F	F	8	...	1	1	1	1	0	0	0
...	F	F	F	9	...	1	1	1	1	0	0	1
...	F	F	F	A	...	1	1	1	1	0	1	0
...	F	F	F	B	...	1	1	1	1	0	1	1
...	F	F	F	C	...	1	1	1	1	1	0	0
...	F	F	F	D	...	1	1	1	1	1	0	1
...	F	F	F	E	...	1	1	1	1	1	1	0
...	F	F	F	F	...	1	1	1	1	1	1	1
				Hexadecimal					Binary			

Signed and Unsigned Number Examples

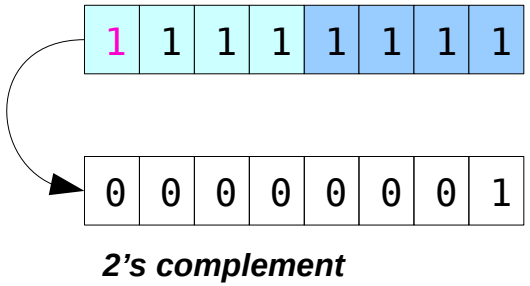
```
unsigned char    u;
```



$$1\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0$$

$$2^8 - 1 = +255$$

```
(signed) char    i;
```



$$= -1$$

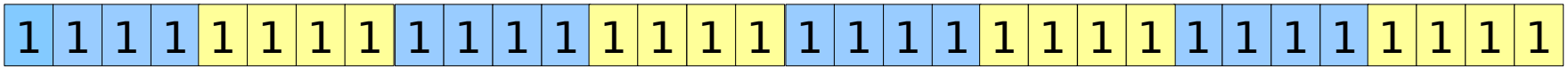
1

0 $\quad 0\cdot2^6 + 0\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot2^0$

$$= +1$$

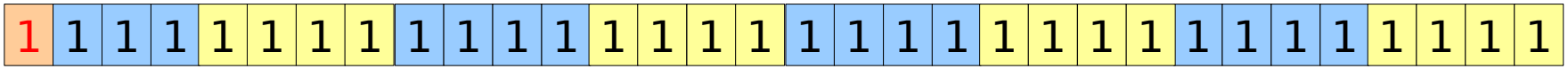
Integer **0xffffffff**

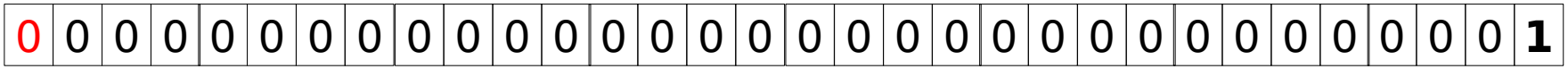
```
unsigned int     u = 0xffffffff;
```



$2^{32} - 1$

```
(signed) int     i = 0xffffffff;
```

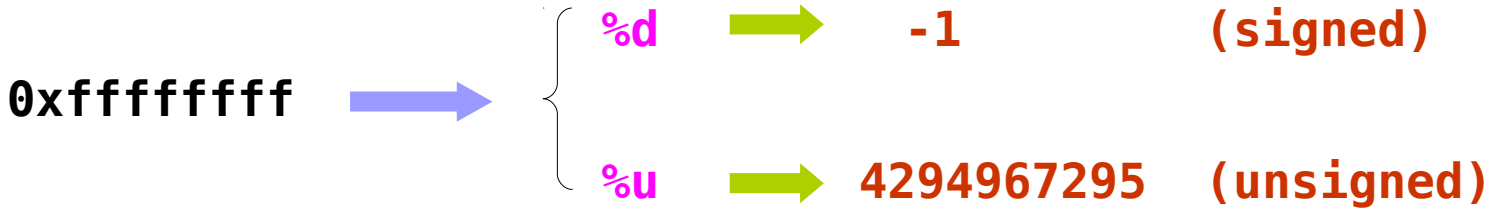




-1

Conversion Specifier %d and %u

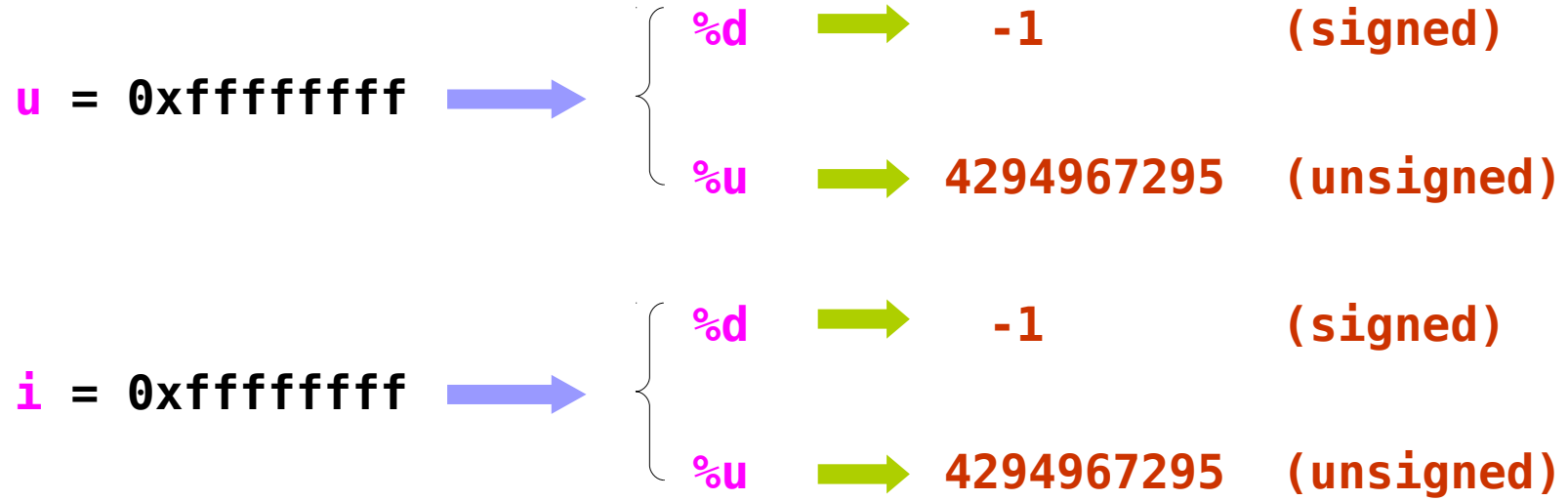
```
unsigned int     u = 0xffffffff;
(signed) int     i = 0xffffffff;
```




```
%d   signed conversion
%u unsigned conversion
```

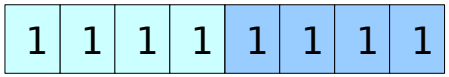
The same results for **4-byte** integer **u** and **i**

```
unsigned int     u = 0xffffffff;
(signed) int     i = 0xffffffff;
```

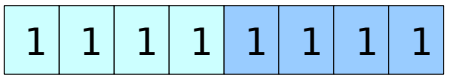


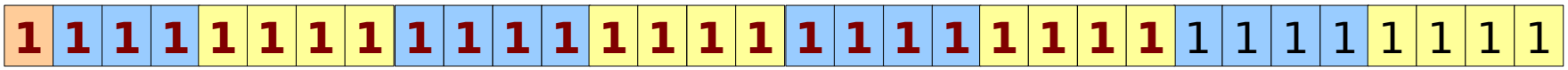
Integer promotion in **printf**

```
unsigned char    u = 0xff;
```



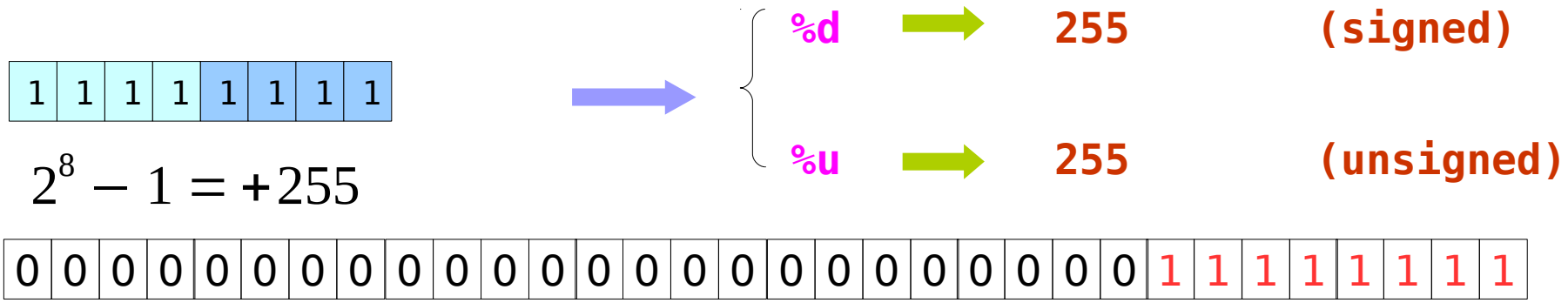

```
(signed) char    i = 0xff;
```



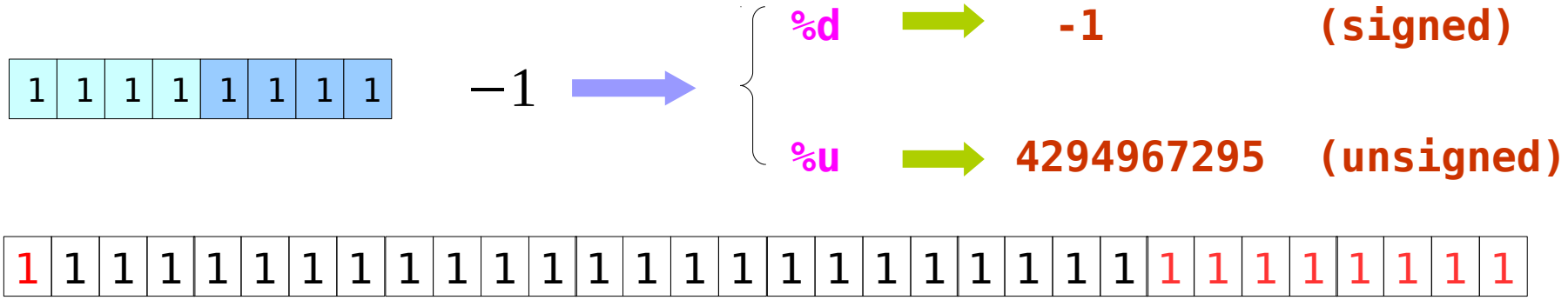


Conversion specifier with **char** type numbers

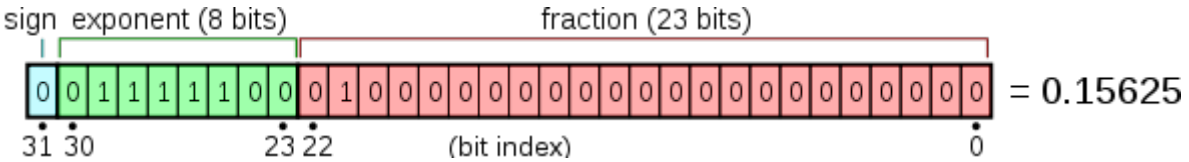
```
unsigned char    u;
```




```
(signed) char    i;
```



Floating Point Number Format

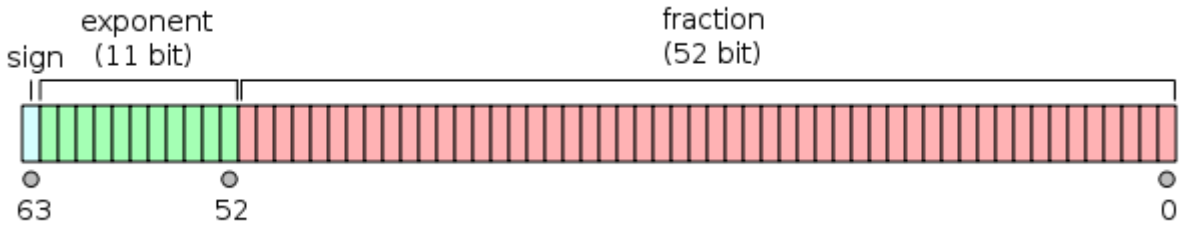


float

- Sign (1-bit)
- Exponent (8-bits)
- Fraction (23-bits)

4 Bytes

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}



double

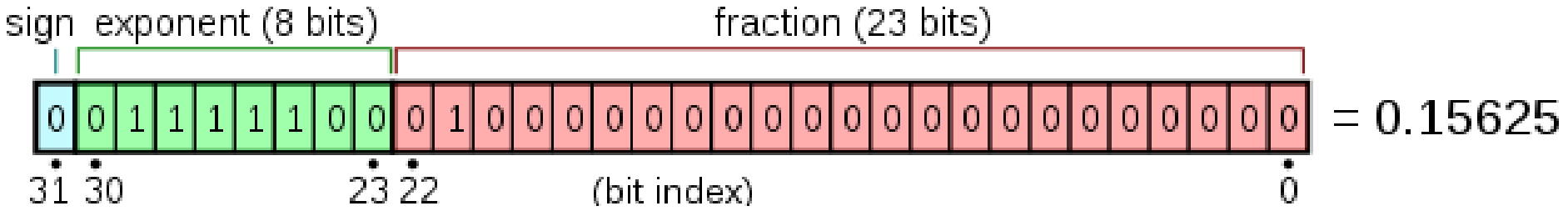
- Sign (1-bit)
- Exponent (11-bits)
- Fraction (52-bits)

8 Bytes

2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

https://en.wikipedia.org/wiki/Single-precision_floating-point_format

Single Precision : float



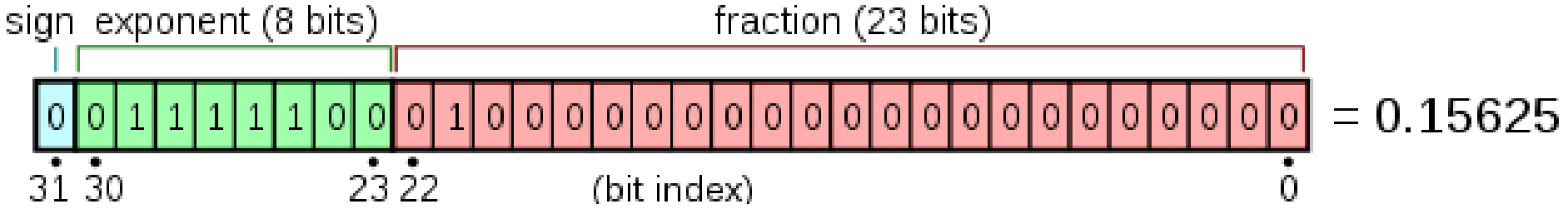
0: positive, 1: negative

exponent Excess +127 (must be subtracted) $127 = 2^{8-1} - 1$

Implicit 1 fraction

https://en.wikipedia.org/wiki/Single-precision_floating-point_format

Single Precision : float

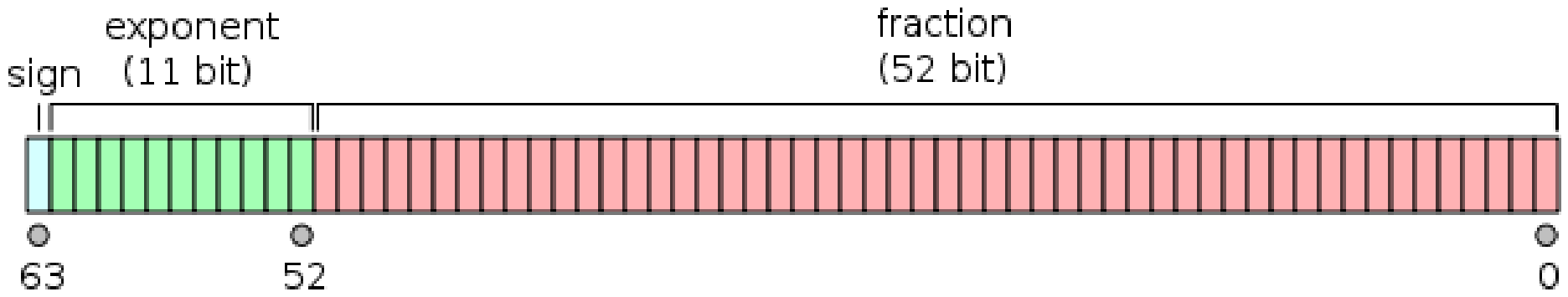


$$(-1)^{b_{31}} \times (1.b_{22}b_{21}\ldots b_0)_2 \times 2^{(b_{30}b_{29}\ldots b_{23})_2 - 127},$$

$$\mathbf{value} = (-1)^{\mathbf{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i}2^{-i}\right) \times 2^{(e-127)}.$$

https://en.wikipedia.org/wiki/Single-precision_floating-point_format

Double Precision : double



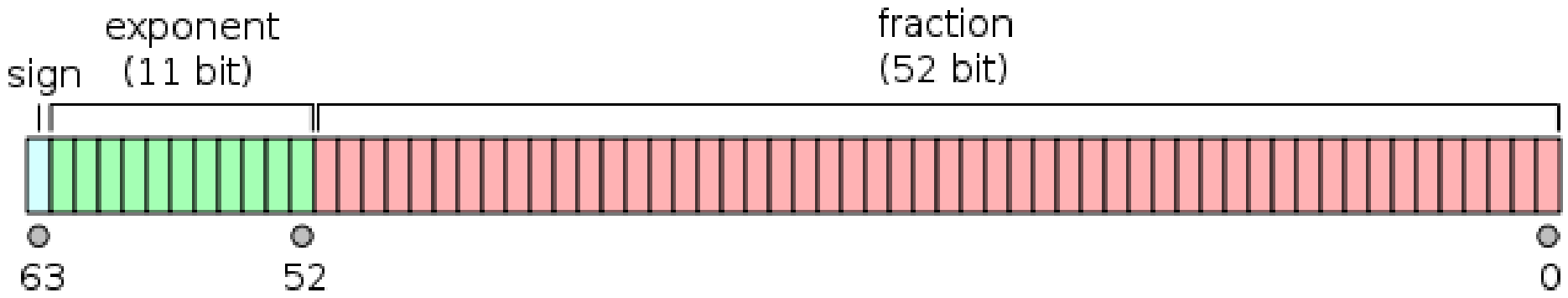
0: positive, 1: negative

exponent Excess +1023 (must be subtracted) $1023 = 2^{11-1} - 1$

Implicit 1 fraction

https://en.wikipedia.org/wiki/Double-precision_floating-point_format

Double Precision : double



$$(-1)^{\text{sign}}(1.b_{51}b_{50}\ldots b_0)_2 \times 2^{e-1023}$$

$$(-1)^{\text{sign}}\left(1+\sum_{i=1}^{52} b_{52-i}2^{-i}\right) \times 2^{e-1023}$$

https://en.wikipedia.org/wiki/Double-precision_floating-point_format

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun