Machine Structure-1

Abdalhafid Boussouf University Center, Mila

2022/2023

The content of the module :

- **Module : Machine Structure 1**
- **Teaching Unit : Fundamental**
- **Credits : 5**
- **Coefficient : 3**

- **Teaching Objectives :**

  The aim of this module is to introduce and deepen the concepts related to different numbering systems as well as the presentation of information, whether it is numeric or character-based.

The basics of Boolean algebra are also covered in-depth.

The content of the module :

Subject Content:

- **Chapter 1 :**

  General Introduction.

- **Chapter 2 : Numbering Systems**

  Definition

- Presentation of systems:

  Decimal, Binary, Octal, and Hexadecimal.

- Conversion between these different systems.
- Basic Operations in the Binary systems :

  Addition, Substraction, Multiplication, Division.

The content of the module :

Chapter 3 : information representation

- **Representation of numbers :**

  1-Integer Numbers : Unsigned Representation , Sign and Absolute Value Representation, One's Complement , Two's Complement.

  2- Fractional numbers : Fixed point, Floating Point.

- **Binary coding:** Pure Binary Code, Reflected Binary Code, Gray Code, DCB code , Excess-3 Code.
- **Character Representation :** EBCDIC Code, ASCII Code, UTF.

The content of the module :

Chapter 4 : Binary Boolean Algebra

- Definition of Boolean Algebra: (Theorems and Properties).
- Logical operators: (AND, OR, negation, NAND and NOR, Exclusive OR) Schematic Representation,
- Truth table, Logical Expressions and Fonctions, Algebraic representation of a function in both the first and second normal forms, Expression of a logical function using NAND or NOR gates..
- Logical function diagram. Simplification of a logical function: (Algebraic Method, Karnaugh Maps, Quine-McCluskey Method)..

Chapter 1 : Numertion Systems

- Introduction
- Information Encoding,
- Numeration Systems
  - The Decimal System
  - The binary System, octal, Hexadecimal
- The transcoding (Base conversion) .
- Arithmetic opérations .

Objectives

- Understanding what information encoding is.
- Learning transcoding (conversion from one base to another).
- Learning to perform arithmetic operations in binary,

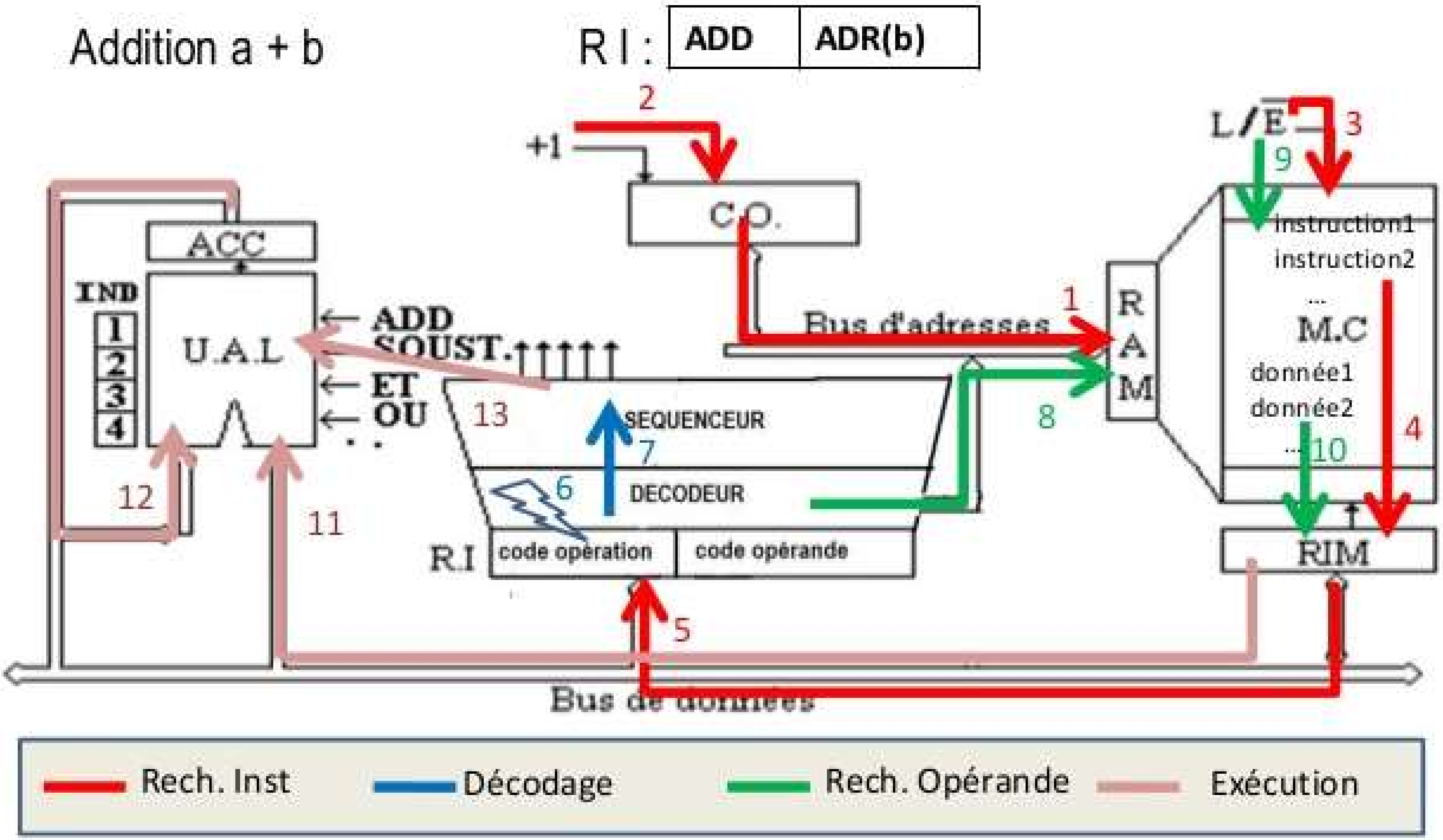
Introduction

- The information processed by computers comes in various forms: numbers, text, images, sounds, videos, programs, ...
- In a computer, they are always represented in binary form (BIT: Binary digIT) as a sequence of 0 s and 1 s.

Processing an instruction

exemple addition (schema)

Addition a + b



Information Encoding :

Definition:

- Encoding allows establishing an unambiguous correspondence between an external representation of information and another representation (called internal), typically in binary form, using a set of precise rules.
- Example: The number 35: 35 is the external representation of the number thirty-five.
- The internal representation of 35 will be a sequence of 0s and 1s (100011).

- We have become accustomed to representing numbers using ten different symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. This system is called the decimal system (decim means ten).
- However, there are other numeral systems that operate using a different number of distinct symbols.
- For example:
  - Binary system (bi: two),
  - The octal system (oct: eight),
  - The hexadecimal system (hexa: sixteen). ...
- In a numeral system, the number of distinct symbols is called the base of the numeral system.

Number systems

Number systems:

- A number system describes how numbers are represented.
- A number system is defined by:

  - A base
  - An alphabet A: a set of symbols or digits
  - Rules for representing numbers

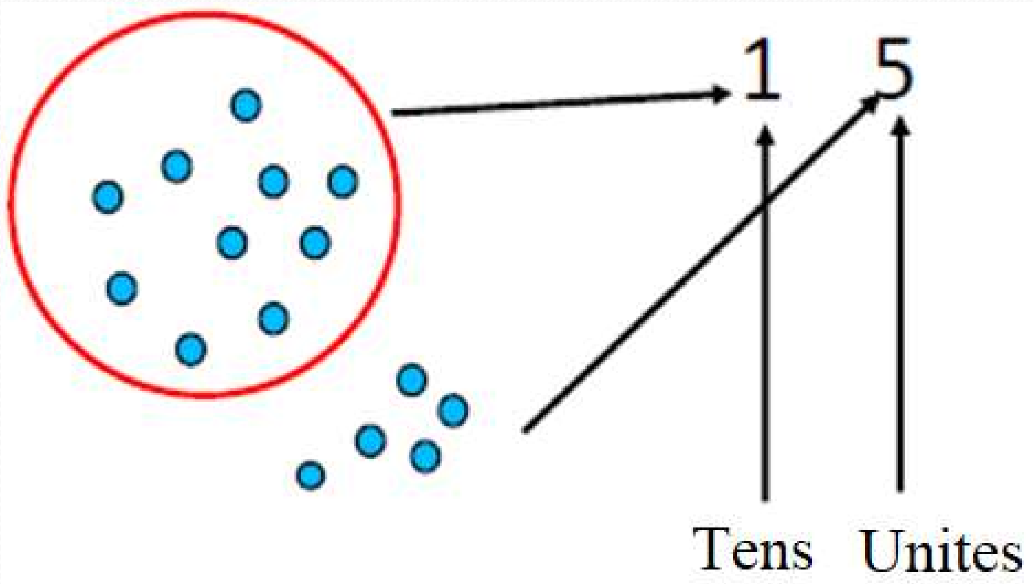
Number Systems:

- The number systems used in the fields of digital electronics and computing are as follows:
    - Binary System (Base 2)
    - Octal System (Base 8)
    - Hexadecimal System (Base 16)

- In addition to the Decimal System (Base 10) used by humans to communicate with the machine.

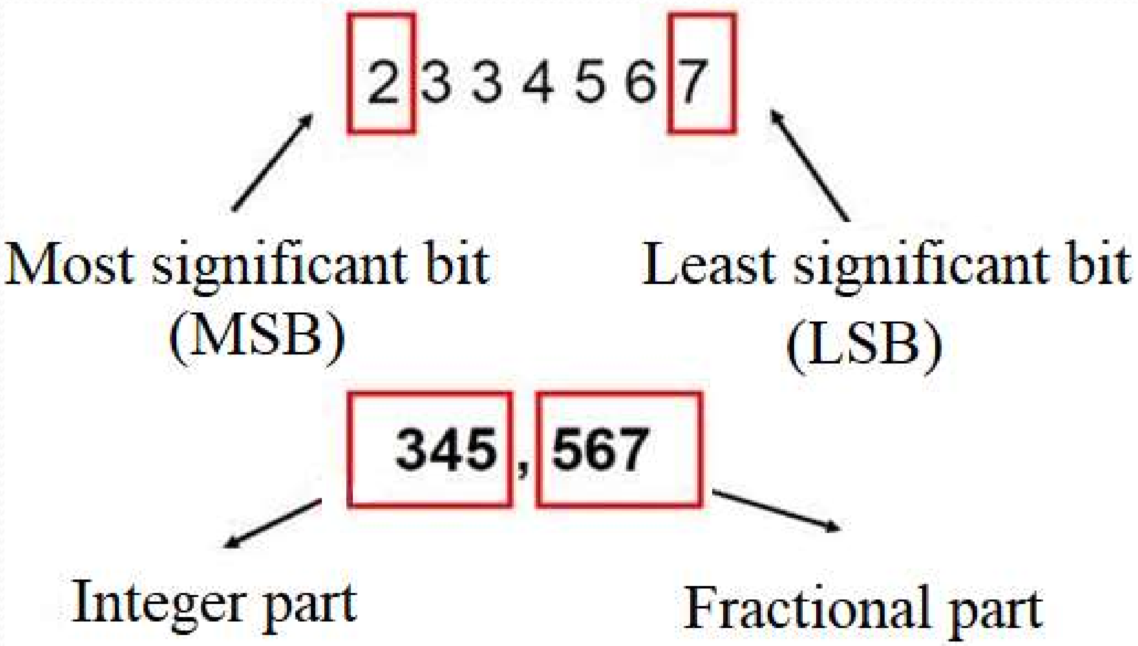
The decimal system:

- Suppose we have 15 tokens, if we form groups of 10 tokens, we will get one group, and there will be 5 tokens left:



The decimal system:

The decimal system's alphabet consists of ten different digits: A = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Any combination of the symbols **{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}** gives us a number.



The decimal system:

- Take the number 1982, this number can be written in the following form:

$(1982)_{10}$=2+80+900+1000 =1*2+8*10+9*100+1*1000

$(1982)_{10}=2*10^0+8*10^1+9*10^2+1*10^3$

This form is called the polynomial form. A real number can also be written in polynomial form.

$(978,265)_{10} = 8*10^0 + 7*10^1 + 9*10^2 + 2*10^{-1} + 6*10^{-2} + 5*10^{-3}$

Decimal counting :

- On a single position: 0 ,1,2,3,4,5,….9= 10^1-1
- On two positions: 00 , 01,02, …..,99=10^2-1
- On three positions : 000,001,……,999=10^3-1

- On n positions:
  - Minimum 0
  - Maximum 10^n-1
  - Number of combinations 10^n

The binary system:

All communication within the computer is done with electrical signals.

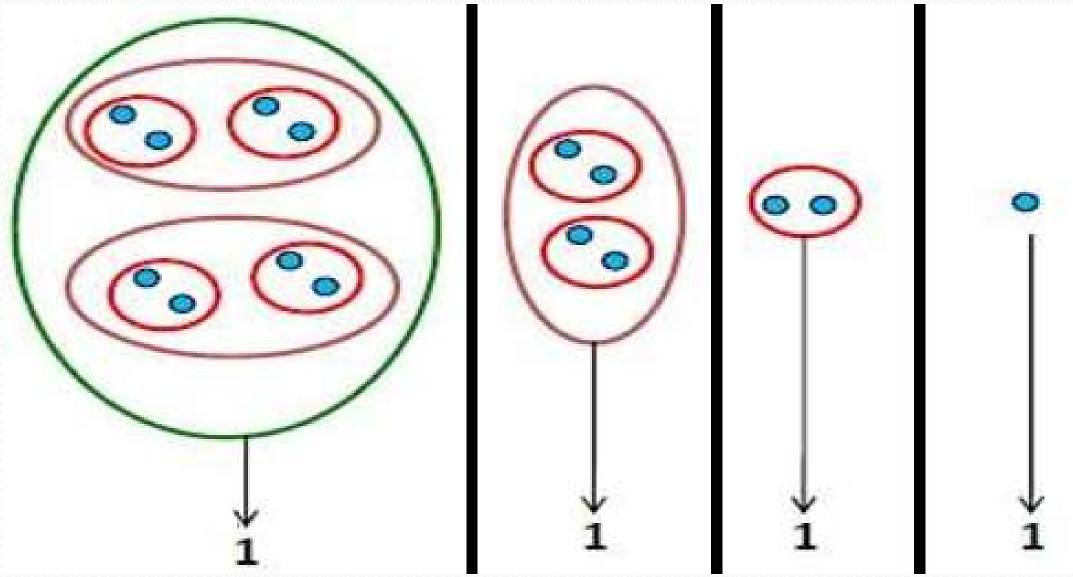
An electrical signal has only two states:

- 1 → absence of an electrical signal
- 2 → presence of an electrical signal

- A unit of information (0 or 1) is called a bit (from the English term 'binary digit').

The binary system :

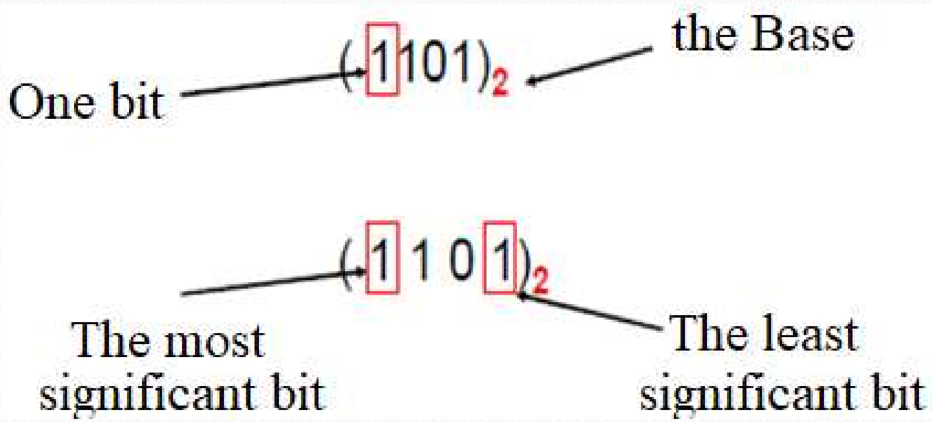
Let's suppose we have 15 tokens, and we form groups of 2 tokens, then continue forming groups of 2 consecutively:



- The number 1111 is the representation of the decimal number '15' in base 2.

The binary system :

In the binary system, to express any value, only 2 symbols are used: A = {0, 1}.



A number in base 2 can also be written in polynomial form.

$(1101)_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3$

$(110,101)_2 = 0*2^0 + 1*2^1 + 1*2^2 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3}$

Binary counting

Exemple

- **on a single bit : 0 , 1**
- **On 2 bits :**

Binary	Decimal
00	0
01	1
10	2
11	3

4 combinaisons= 2^2

On 3 Bits

$2^2 2^1 2^0$

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

8 combinaisons= 2^3

On 4 Bits

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

16 combinaisons= 2^4

The Binary Counting:

- With one bit, we can encode two states; with 2 bits, four states... With each new bit, the number of possible combinations doubles.
- Using n bits, we can form 2^n different numbers, and the largest among them is equal to (2^n-1). For example, if n= 8, Nmax = (2^8-1) = 255, We can form 256 different numbers from:

  $(0)_{10} = (00000000)_2$ -to- $(255)_{10} = (11111111)_2$
- Note: A group of eight bits is called a byte

The octal system:

It is the base-8 system

Eight (8) symbols are used in this system:

A={ 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 }

- Example of polynomial form :

$(237)_8 = 7*8^0 + 3*8^1 + 2*8^2$

$(53,948)_8 = 3*8^0 + 5*8^1 + 9*8^{-1} + 4*8^{-2} + \quad 8*8^{-3}$

- **Exemple 2 :**

The number (1289) does not exist in base 8 since the symbols 8 and 9 do not belong to the octal base.

The hexadecimal system:

It is the base-16 system

Sixteen different symbols are used:
{0,1,2,3,4,5,6,7,8,9,A, B, C, D,E, F}

- **Exemple of polynomial form:**

$(A4C)_{16} = 12*16^{0} + 4*16^{1} + 10*16^{2}$

$(14,2B)_{16} = 4*16^{0} + 1*16^{1} + 2*16^{-1} +11*16^{-2}$

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Generalization: System B:

- In a base B,
- B distinct symbols are used to represent numbers.
- The value of each symbol must be strictly less than the base B.
- Every number in a base B can be written in its polynomial form.

$N_B = (a_{n-1}\, a_{n-2} \ldots\, a_2\, a_1\, a_0,\ \mathbf{a_{-1} \ldots\, a_{-m}})_B = a_0.B^0 + \ldots + a_{n-2}.B^{n-2} + a_{n-1}.B^{n-1} + \mathbf{a_{-1}.B^{-1}} + \ldots + \mathbf{a_{-m}.B^{-m}} = N_{10}$

Transcoding:
(Base Conversion)

Definition of transcoding :

- Transcoding (or base conversion): is the operation that allows us to switch from the representation of a number expressed in one base to the representation of the same number but expressed in another base.

- Next, we will see the following base conversions:

- Decimal to Binary, Octal, and Hexadecimal
- Binary to Decimal, Octal, and Hexadecimal

Conversion from base B to base 10:

- This conversion is quite simple because all you need to do is to: expand this number in polynomial form in base B and then add them up.

- **Exemples :**

- $(1101)_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 = (13)_{10}$
- $(1A7)_{16} = 7*16^0 + 10*16^1 + 1*16^2 = (423)_{10}$
- $(1101,101)_2 = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = (13,625)_{10}$

- $(43,2)_5 = 3*5^0 + 4*5^1 + 2*5^{-1} = (23,4)_{10}$

Conversion from decimal to base B:

- **Principle:**

- **Division** by **B** for the **integer part**.

- **Multiplication** by **B** for the **fractional part**.

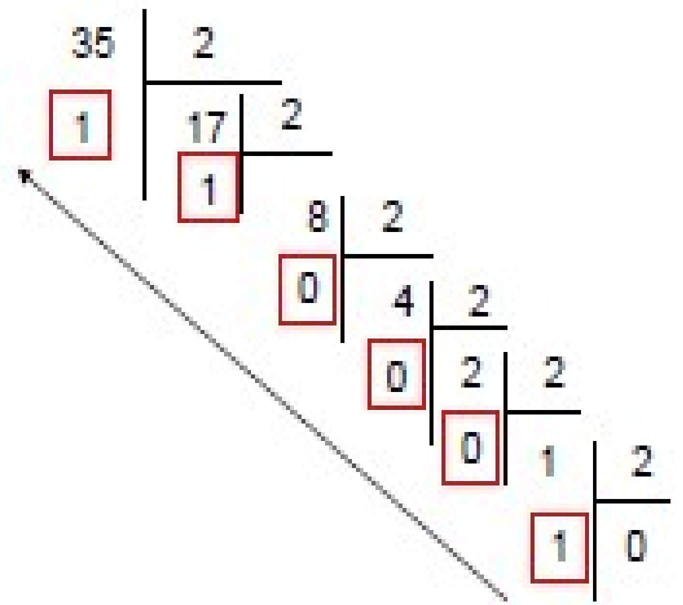
Conversion from base 10 to base 2:

- The principle consists of performing successive divisions of the number by 2, and taking the remainders of the divisions in reverse order.

- Exemple : $(35)_{10}=(?)_2$

- After division : →

- you obtain :$(35)_{10} =$ **$(100011)_2$**



Conversion from base 10 to base 2:

- (in the case of a real number): A real number consists of two parts:
- The integer part and the fractional part. The integer part is transformed by performing successive divisions.
- The fractional part is transformed by performing successive multiplications by 2.
- **Exemple 1:** 35,625=(?)2

0,625 * 2 =**1**,250

0,25* 2 = **0** ,50

0,5* 2 = **1** ,0,

↓

I.P =35 = (100011)2

F.P= 0,625) = (0,101)2

Donc 35,625=(100011,101)2

Conversion from base 10 to base 2:

- **Exemple 2:**
- Perform the following conversion, $(0,7)_{10}=(?)_2$

(in the case of a real number)

0,7 * 2 = 1,4
0,4 * 2 = 0,8
0,8 * 2 = 1,6
0,6 * 2 = 1,2
0,2 * 2 = 0,4

$(0,7)= (0,10110)_2$

- The number of bits after the decimal point will determine the precision.

Conversion from base 10 to base 2:

- **Note:** Sometimes, when multiplying the decimal part by Base B, we may not be able to convert the entire integer part. This is mainly due to the fact that the number being converted does not have an exact equivalent in Base B, and its decimal part is cyclic.
- 0,15*2=0,3
- 0,3*2=0,6
- 0,6*2=1,2
- 0,2*2=0,4
- 0,4*2=0,8
- 0,8*2=1,6
- 0,6*2=1,2
- The result is, therefore : (0,15)10= (0, 0010011001...)2. It is said that (0,15)10 is cyclic in Base 2 with a period of 1001.
- **Note:** After several multiplications, we stop the calculations.

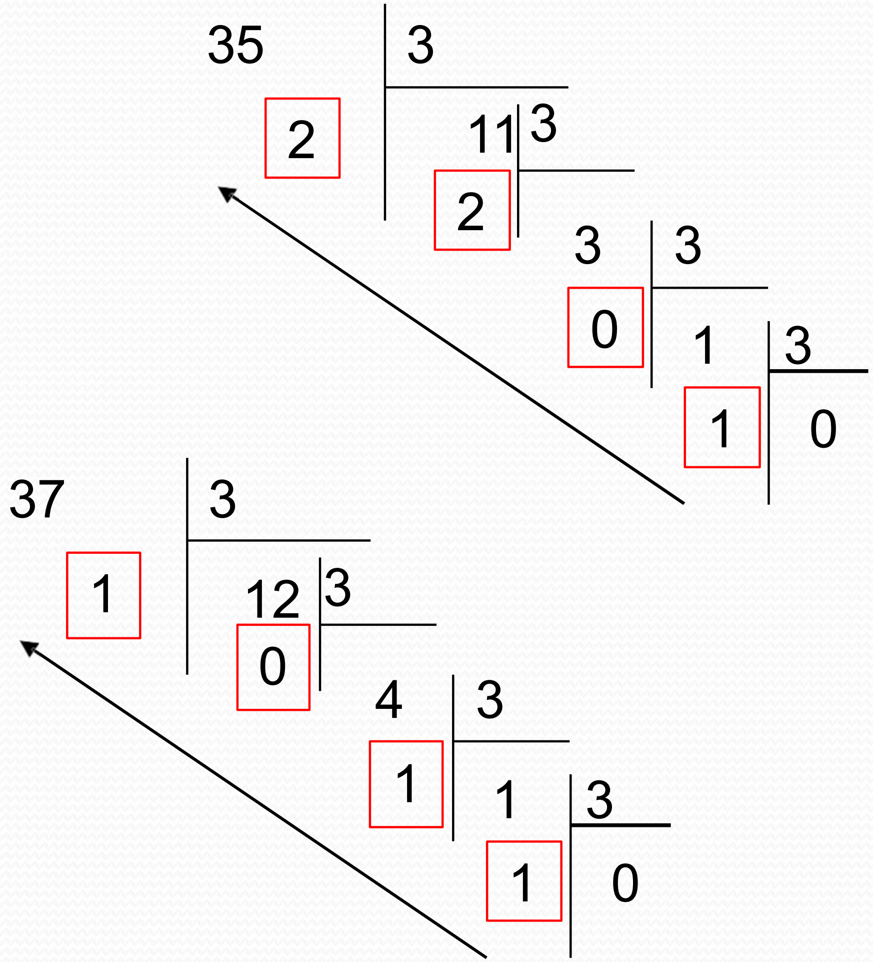
Conversion from decimal to base X

- The conversion is done by taking the remainders of successive divisions in base X in reverse order.

Example: Perform the following conversions:

$(35)_{10} = (?)_3$

$(37)_{10} = (?)_3$

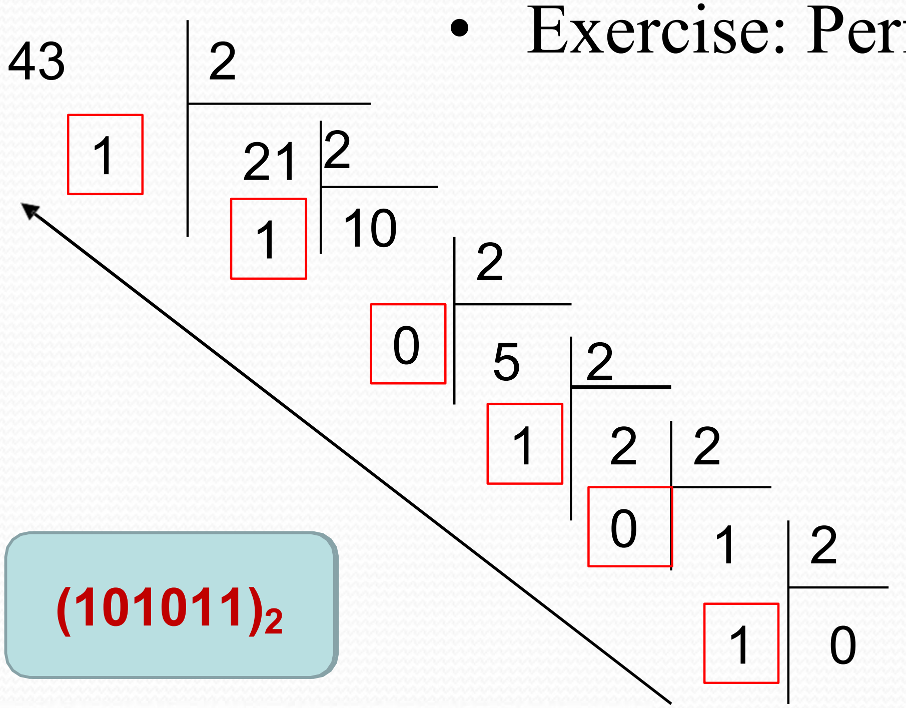


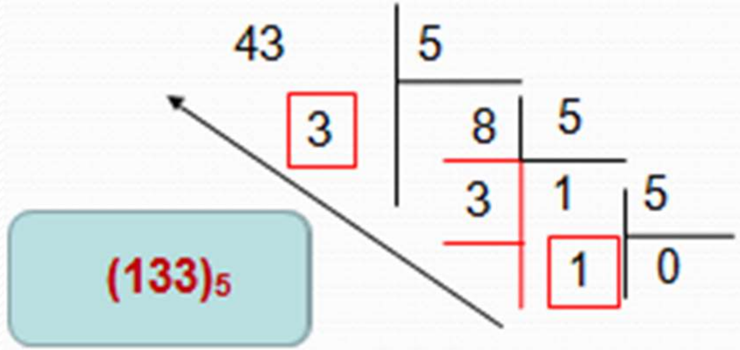
$(35)_{10} = (1022)_3$
$(37)_{10} = (1101)_3$

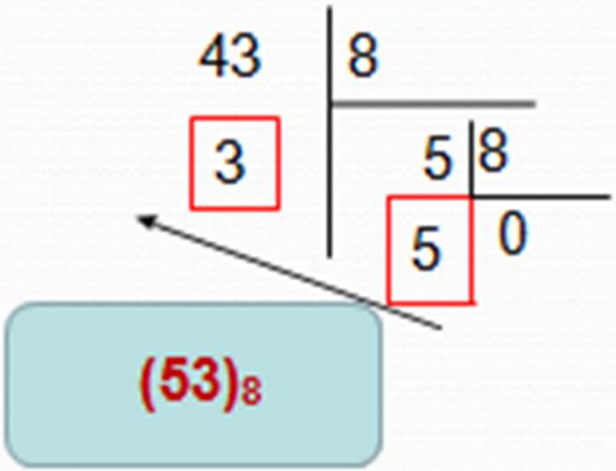
Conversion from decimal to base X

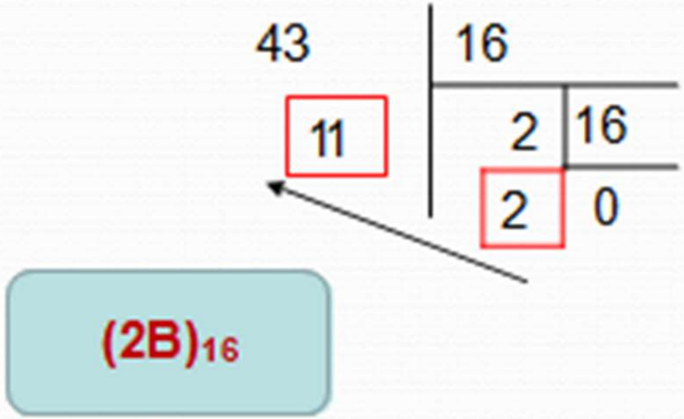
- Exercise: Perform the following transformations.

$$(43)_{10}=(?)_2=(?)_5=(?)_8=(?)_{16}$$



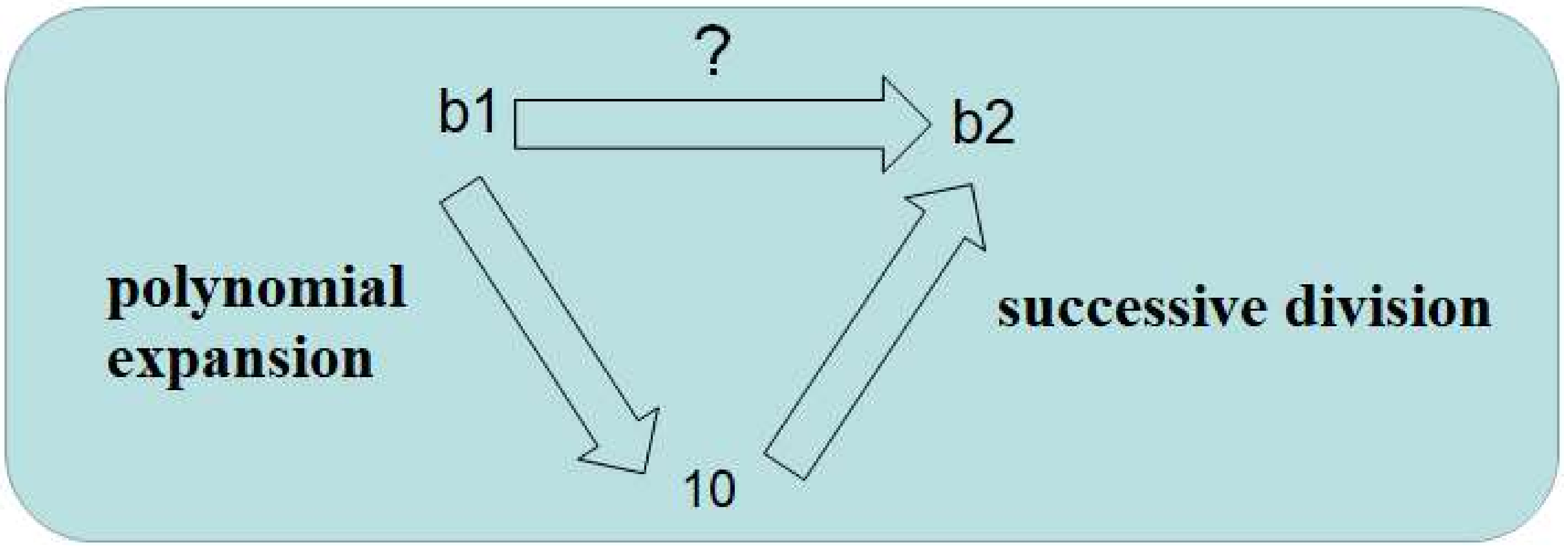






Conversion from base b1 to base b2

- To go from one base **b1** to another base **b2** directly (usually there is no method!!)
- The idea is to convert the number from base **b1** to base **10**, and then convert the result from base **10** to base **b2**.

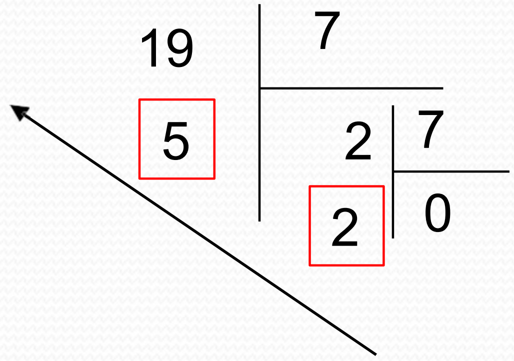


Conversion from base b1 to base b2

Exercise: Perform the following conversion.

$(34)_5=(?)_7$

$$(34)_5 = 3*5^1 + 4*5^0 = 15 + 4 = (19)_{10} = (?)_7$$



$(34)_5$ $=(19)_{10}=(25)_7$ ⇒ $(34)_5=(25)_7$

Octal to binary conversion :

- In octal, each symbol of the base is represented by 3 bits in binary. The basic idea is to replace each symbol in the octal base with its 3-bit binary value (performing expansions into 3 bits).
- **Exemples :** $(345)_8$=?; $(65,76)_8$=?; $(35,34)_8$=?

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- $(345)_8 = (\underline{011}\ \underline{100}\ \underline{101})_2$
- $(65,76)_8 = (\underline{110}\ \underline{101},\ \underline{111}\ \underline{110})_2$
- $(35,34)_8 = (\underline{011}\ \underline{101}\ ,\ \underline{011}\ \underline{100})_2$

- **Note**: The replacement is done from right to left for the integer part and from left to right for the fractional part.

Octal to binary conversion :

- The basic idea is to group the bits in sets of 3 starting from the least significant bit.
- Then, replace each group with the corresponding octal value.

- **Exemples :** $(11001010010110)_2$ =?; $(110010100,10101)_2$ =?

$(11001010010110)_2 = (\underline{011}\ \underline{001}\ \underline{010}\ \underline{010}\ \underline{110})_2 = \mathbf{(31226)_8}$

$(110010100,10101)_2 = (\underline{110}\ \underline{010}\ \underline{100}\ ,\ \underline{101}\ \underline{010})_2 = \mathbf{(624,52)_8}$

<---   ---->

- **Note:** Grouping is done from right to left for the integer part and from left to right for the fractional part.

Hexadecimal to binary conversion:

- In hexadecimal, each symbol in the base is represented using 4 bits. The basic idea is to replace each symbol with its 4-bit binary value (performing splitting into 4 bits).

- **Exemples :**$(757F)_{16}$=? ; $(BA3,5F7)_{16}$=?

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

$(757F)_{16}=(\underline{0111}\ \underline{0101}\ \underline{0111}\ \underline{1111})_2$

$(BA3,5F7)_{16}=(\ \underline{1011}\ \underline{1010}\ \underline{0011}\ ,\ \underline{0101}\ \underline{1111}\ \underline{0111}\ )_2$

Hexadecimal to binary conversion:

- The basic idea is to group the bits into sets of 4, starting from the least significant bit. Then, replace each group with the corresponding hexadecimal value.
- **Exemples :**
- $(11001010100110)_2$=?
- $(110010100,10101)_2$ = ?

$(11001010100110)_2=(0011\ 0010\ 1010\ 0110)_2=(32A6)_{16}$

$(110010100,10101)_2= (0001\ 1001\ 0100,1010\ 1000)_2=(194,A8)_{16}$

"Conversion from base B1 to base B2

- Both bases are powers of 2 (base 8 and 16).

The base 2 is used as an intermediate base.

- Base B1 → Base 2 → BaseB2

- Both bases are not powers of 2.

The base 10 is used as an intermediate base.

- Base B1 → Base 10 → Base B2

Arithmetic operations

Arithmetic operations

- **General principle:**

- The rules of arithmetic operations in decimal are also valid for arithmetic operations in any base.

Binary Addition

- To add two binary numbers, we proceed exactly as in decimal, but taking into account the following elementary addition table:

- 0+0 = 0 carry 0
- 0+1 = 1 + 0 = 1 carry 0
- 1 + 1 = 0 carry 1
- 1 + 1 + 1 = 1 carry 1

Exemple :

	In binary									In decimal			
Carry →		*1*	*1*	*1*	*1*	*1*	*1*	*1*				*1*	
			1	1	1	0	0	1	1		1	1	5
	+				1	1	1	0	1	+		2	9
Result →	=	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	=	**1**	**4**	**4**

Binary Addition

- **Exemple:**

a	*b*	*S*	*R*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

S : The sum

R : The carry

						1	1	
		1	1	0	0	0	1	1
+	1	0	0	0	1	0	1	1
	1	1	1	0	1	1	1	0

Arithmetic operations in binary

- **Exercice:** Perform the following operation

$(\mathbf{1100011})_2 + (\mathbf{10001011})_2 = (?)_2$:

0 + 0 = 0	0 + 1 = 1	1 + 0 = 1	1 + 1 = 10

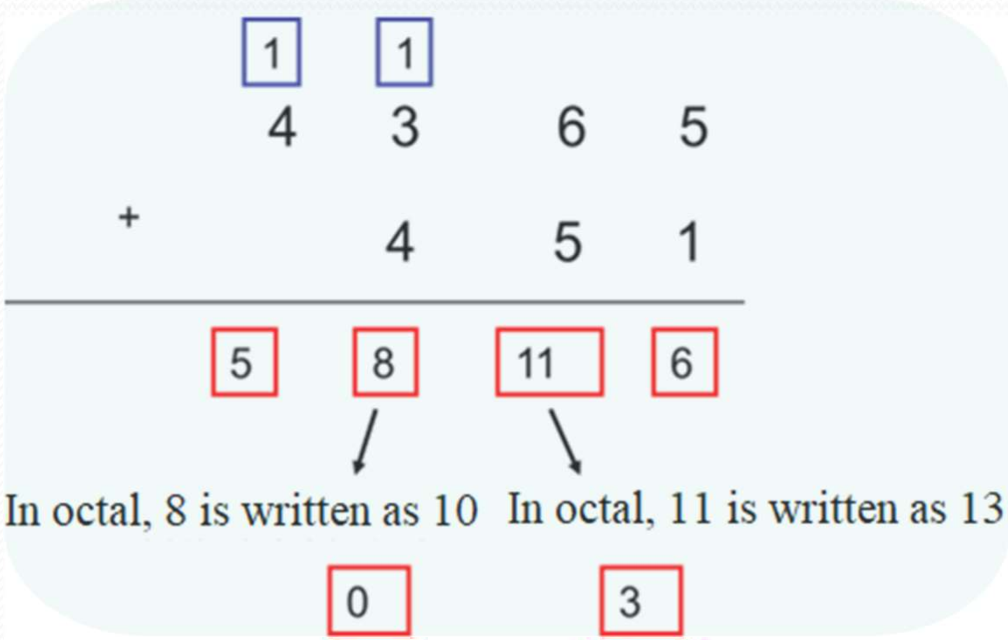
The resultat : $(\mathbf{11101110})_2$

					1	1		
		1	1	0	0	0	1	1
+	1	0	0	0	1	0	1	1
	1	1	1	0	1	1	1	0

Arithmetic operations in octal

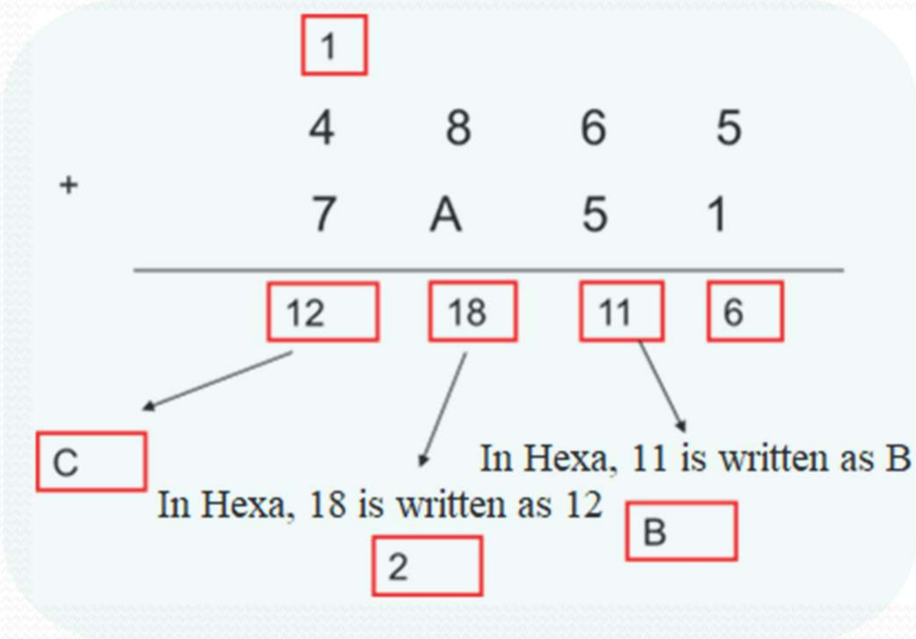
Exercice: Perform the following operation :

$(4365)_8 + (451)_8 = (?)_8$ $(4865)_{16} + (7A51)_{16} = (?)_{16}$



Octal Operations

The result : $\mathbf{(5036)_8}$



Hexadecimal Operations

The result : $\mathbf{(C2B6)_{16}}$

binary subtraction

- In binary subtraction, when the quantity to subtract is greater than the quantity being subtracted from, we borrow 1 from the left neighbor. In binary, this 1 adds 2 to the quantity being subtracted from, while in decimal, it adds 10.

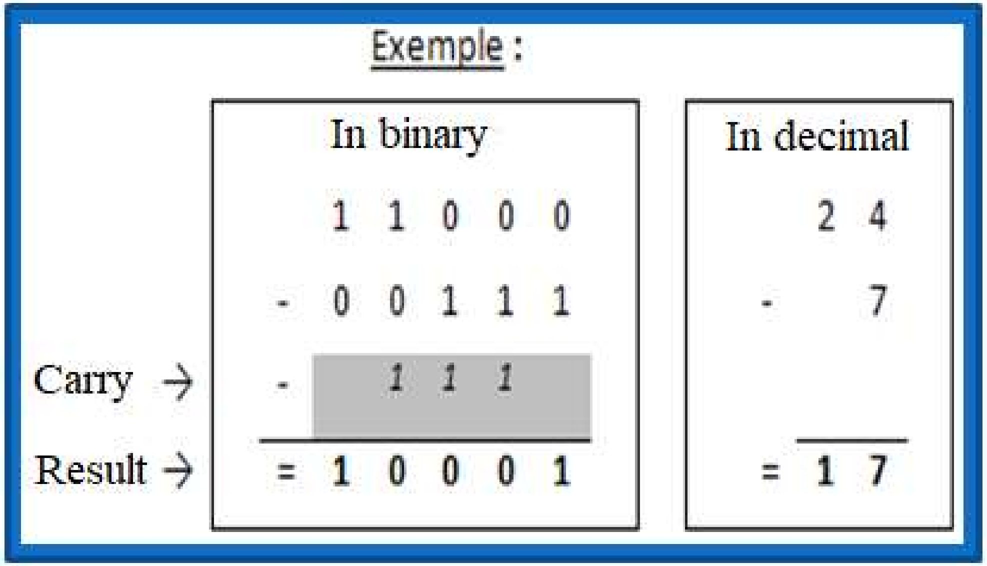
Table de soustraction binaire :

- 0 - 0 = 0 Carry 0
- 1 - 0 = 1 Carry 0
- 0 - 1 = 1 Carry 1 is subtracted

from the left neighbor digit

- 0 - 1 - 1 = 0 Carry 1 is subtracted

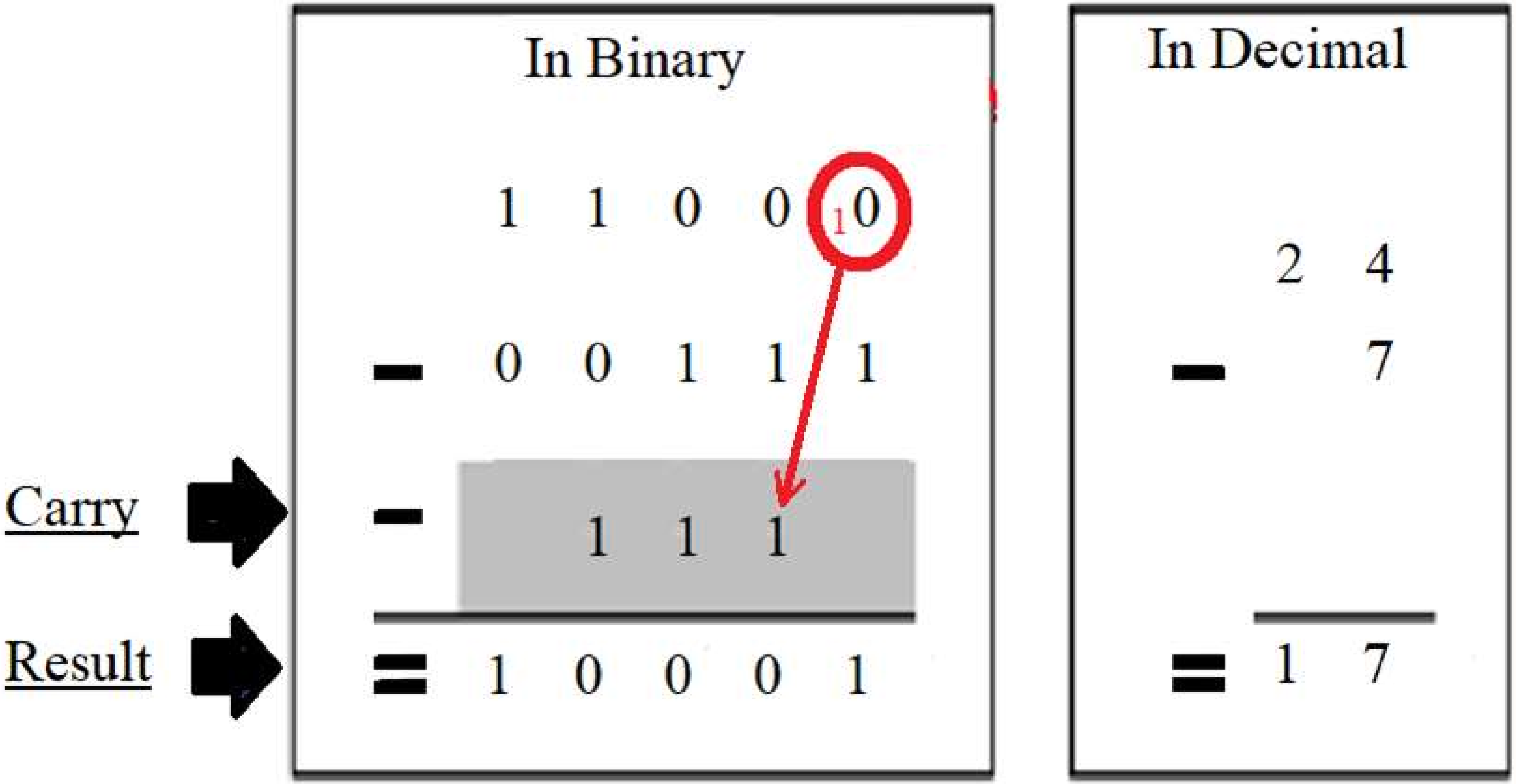
from the left neighbor digit



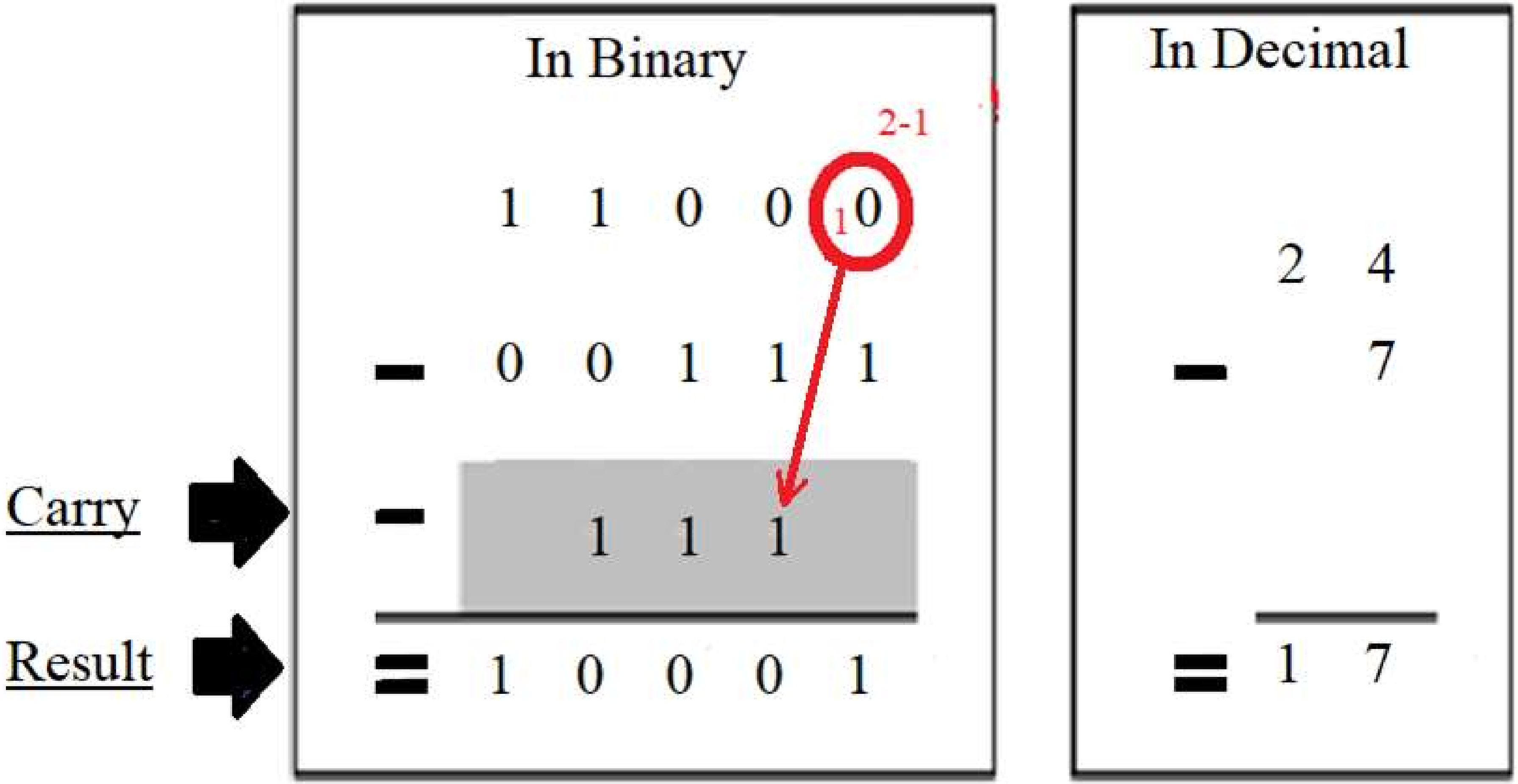
Exemple:

	In Binary						In Decimal	
		1	1	0	0	0		2 4
	−	0	0	1	1	1	−	7
Carry ➡	−							
Result ➡	=						=	

Exemple:

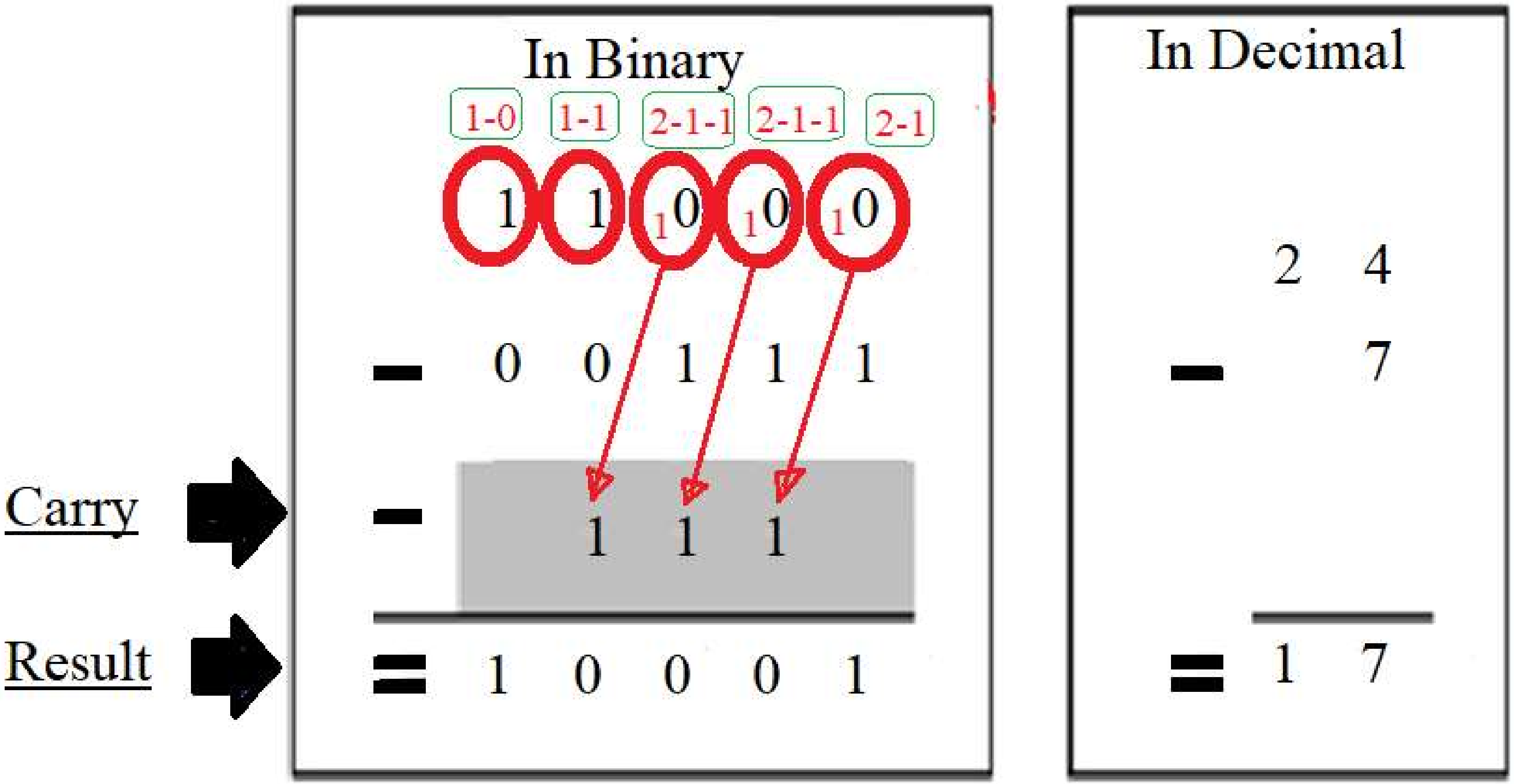

In Binary
1 1 0 0 $_1$0
– 0 0 1 1 1
Carry
– 1 1 1
Result
= 1 0 0 0 1
In Decimal
2 4
– 7
= 1 7

Exemple:



In Binary
2-1
1 1 0 0 $_1$0
− 0 0 1 1 1
Carry
− 1 1 1
Result
= 1 0 0 0 1
In Decimal
2 4
− 7
= 1 7

Exemple:


In Binary
1-0
1-1
2-1-1
2-1-1
2-1
1 1 0 0 0
– 0 0 1 1 1
Carry
– 1 1 1
Result
= 1 0 0 0 1
In Decimal
2 4
– 7
= 1 7

binary subtraction

- Exemple

a	*b*	*D*	*E*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

D : Difference
E : Borrowed

```
  1 0 1 1 1 0 1 1 0 0 0
- 0 0 0 0 1 1 0 0 1 1 1
  _____________________
```

binary subtraction

- Exemple

a	*b*	*D*	*E*
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

D : Difference

E : Borrowed

```
   1 0 1 1 1 0 1 1 0 0 0
 - 0 0 0 0 1 1 0 0 1 1 1
 -----------------------
   1 0 1 0 1 1 1 0 0 0 1
```

Binary Multiplication

- Binary multiplication is performed in a similar way to decimal multiplication. Here are the calculation rules to use:

- 0 x 0 = 0
- 0 x 1 = 0
- 1 x 0 = 0
- 1 x 1 = 1

It consists of making a series of additions with the multiplicand shifted to the left. this operation is repeated as many times as there are binary elements (at 1) in the multiplier.

- Note: When an operation results in more than two partial products, add these products together in pairs to reduce the risk of errors.

Exemple :

1101	Multiplicand
x 1011	Multiplier
0001101	
+ 001101	Shift 1 step
+ 1101	Shift 3 steps
10001111	result

Binary Multiplication

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

$c = a \cdot b$

- Exemple

```
    1 0 1 1 0 0 1 0 1
x           1 0 1 1
  -----------------
```

Binary Multiplication

a	*b*	*c*
0	0	0
0	1	0
1	0	0
1	1	1

c = a . b

- Exemple

```
              1 0 1 1 0 0 1 0 1
  x                     1 0 1 1
              -----------------
              1 0 1 1 0 0 1 0 1
            1 0 1 1 0 0 1 0 1
          0 0 0 0 0 0 0 0 0
        1 0 1 1 0 0 1 0 1
        -----------------------
        1 1 1 1 0 1 0 1 0 1 1 1
```

Binary Multiplicaion

- Exemple

			1	0	1	0
x	1	0	1	0	1	0

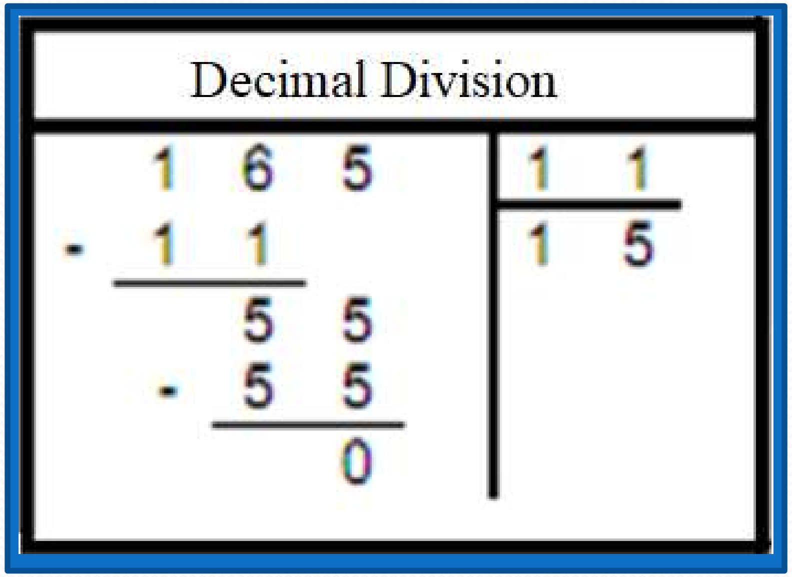
Binary Multiplication

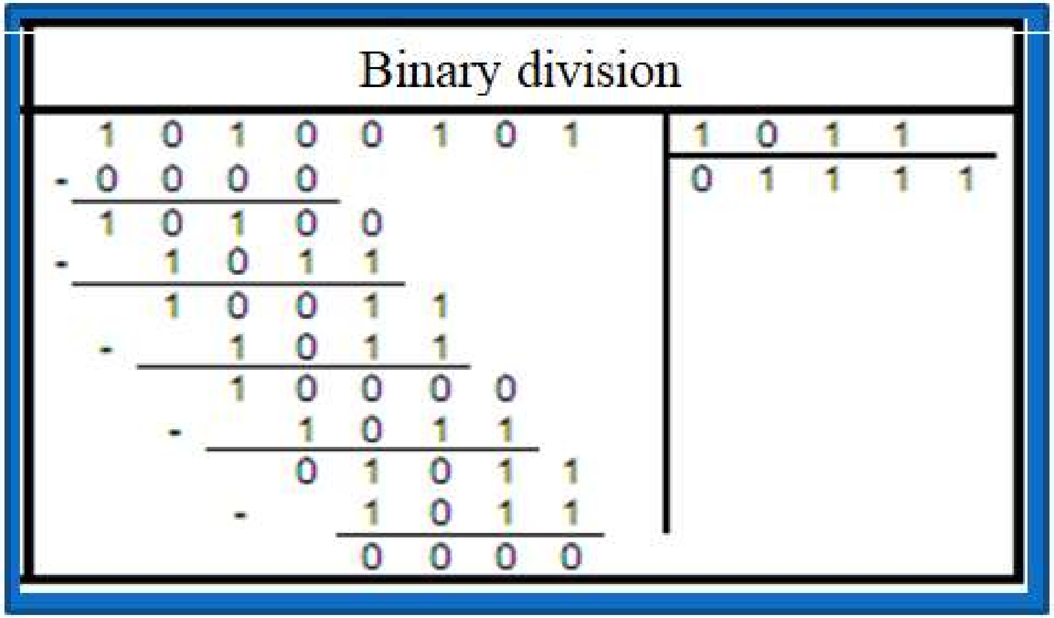
- Exemple

						1	0	1	0
x				1	0	1	0	1	0
						0	0	0	0
+				(1)	1	0	1	0	0
+				0	0	0	0	0	0
+		(1)	1	0	1	0	0	0	0
+		0	0	0	0	0	0	0	0
+	1	0	1	0	0	0	0	0	0
=	1	1	0	1	0	0	1	0	0

binary Division

- Binary division is performed using subtractions and shifts, similar to decimal division, except that the quotient digits can only be 1 or 0. The quotient bit is 1 if the divisor can be subtracted, otherwise it is 0.





binary Division

Exemple

```
  1 1 1 0 0 1 1 1      | 1 0 1
- 1 0 1                |-----------
  -------              |
  0 1 0 0 0            | 1 0 1 1 1 0
-     1 0 1            |
    --------           |
    0 0 1 1 1          |
-         1 0 1        |
      ---------        |
          0 1 0 1      |
-           1 0 1      |
       ----------      |
            0 0 0 1    |
-             0 0 0    |
         ----------    |
              0 0 1    |
```

111100111 : 101 = 101110   reste 1.

binary Division

Exemple

```
1 1 1 0 1 | 1 0 1
          |------
1 0 1     | 1 0 1
-----     |
  1 0 0 1 |
    1 0 1 |
  --------|
    1 0 0 |
```

In fact, simply subtract 101 when possible, and lower the following number: 11101=101*101+100

binary Division

Exemple

1 0 1 1	100

binary Division

Exemple

```
  1 0 1 1     | 100
 -1 0 0       |---------
  -------     | 10,11
  0 0 1 1     |
 - 0 0 0      |
  -------     |
      1 1 0   |
     -1 0 0   |
    -------   |
        1 0 0 |
       -1 0 0 |
      ------- |
            0 |
```

Arithmetic Operations in Octal:

- **The addition**
- Just like in the binary system, the same rules apply to octal numbers. However, in this case, we will have a carry-over '1' to the left whenever the sum exceeds the value of 7 because 7(8) + 1(8) = 10(8).
- Example of addition in octal base:

$$
\begin{array}{cccc}
 & {}^{1}1 & {}^{1}2 & 3_{(8)} \\
+ & 4 & 5 & 7_{(8)} \\
\hline
 & 6 & 0 & 2_{(8)}
\end{array}
$$

Arithmetic Operations in Octal:

- **Subtraction**
- Similar to decimal subtraction but limited to 7 (octal).
- Exemple:

$$
\begin{array}{lccc}
 & 1 & 2 & 3_{(8)} \\
- & 4 & 5 & 7_{(8)} \\
\hline
 & & & _{(8)}
\end{array}
$$

Arithmetic Operations in Octal:

- **Subtraction**
- Similar to decimal subtraction but limited to 7 (octal).
- Exemple:

	${}^{1}1$	${}^{1}2$	${}_{1}3_{(8)}$
-	${}_{1}4$	${}_{1}5$	$7_{(8)}$
	4	4	$4_{(8)}$

Hexadecimal Arithmetic Operations

- **Addition**
- Similar to decimal addition, hexadecimal addition is performed digit by digit. However, in this case, there will be a carry '1' to the left each time the sum exceeds the value F because: $\mathbf{(F_{16}+1_{16})=10_{(16)}}$.

$$
\begin{array}{cccc}
 & 1 & {}^{1}2 & A_{(16)} \\
+ & E & 5 & 7_{(16)} \\
\hline
 & F & 8 & 1_{(16)}
\end{array}
$$

Hexadecimal Arithmetic Operations

- **The substraction in hexadecimal:**
- **Exemple:**

	F	2	$A_{(16)}$
-	E	5	$7_{(16)}$
			$_{(16)}$

Hexadecimal Arithmetic Operations

- **The substraction in hexadecimal :**
- **Exemple:**

	F	$^{1}2$	$A_{(16)}$
-	$_{1}E$	5	$7_{(16)}$
	0	D	$3_{(16)}$

Hexadacimal Multiplication

Example

Find multiplication of

$(B84F)_{16}$

x $(A53)_{16}$

Solution:

		7	5	3	9	
		3	2	1	4	
		2	1		2	
			B	8	4	F
			×	A	5	3
		2	2	8	E	D
	3	9	9	8	B	0
7	3	3	1	6	0	0
7	6	E	D	7	9	D

3 D E 5 | A

Hexadecimal

Division

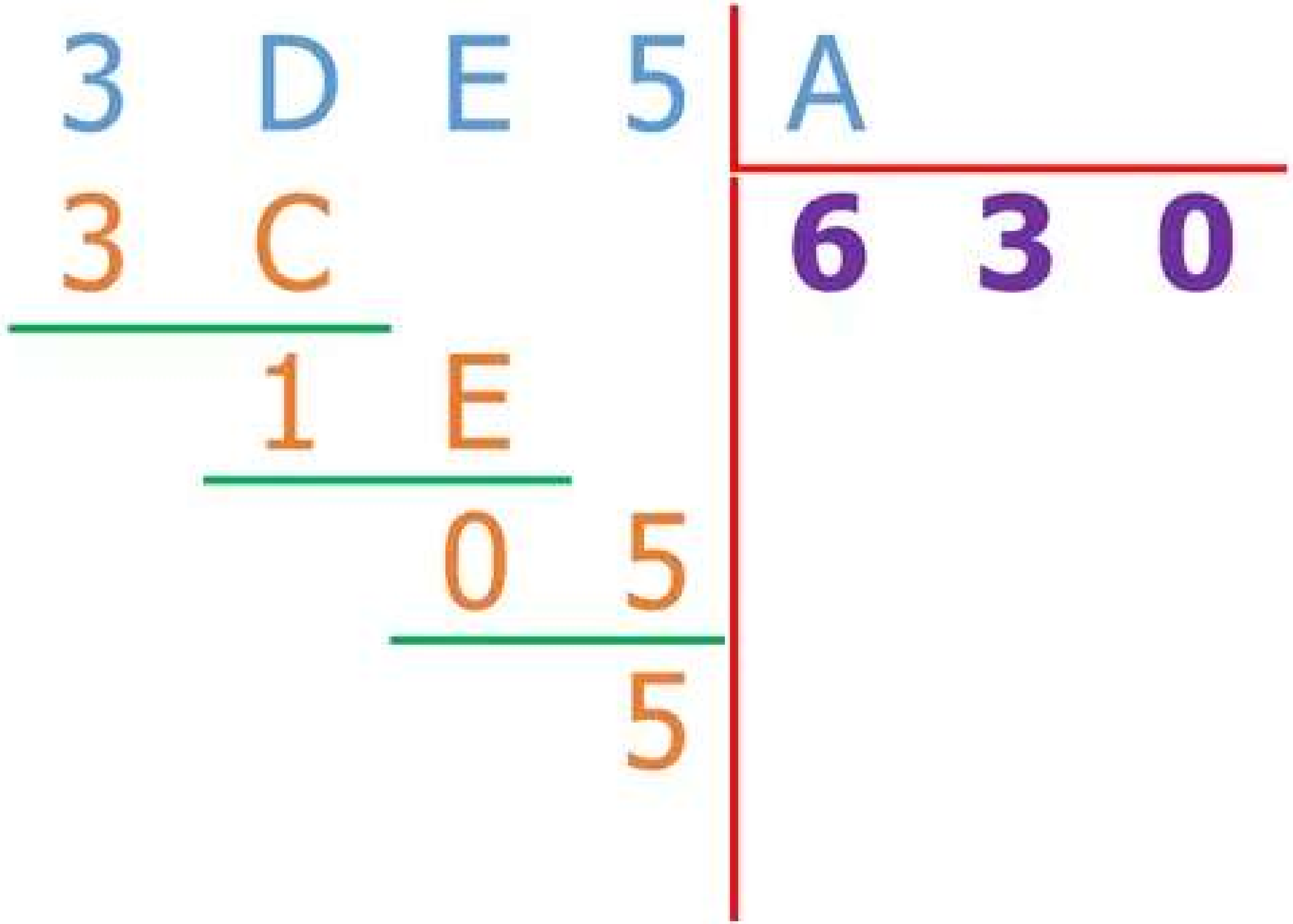
Exemple: find division of:

3DE5 / A

3 D E 5 | A

6

3	D	E	5	A
3	C			6 3 0
	1	E		
		0	5	
			5	



$$3DE5_{16} \div A_{16} = 630_{16}$$

Thanks a lot