Chapter 2: Number System

1. Definition

- A Number System is a systematic way of representing and expressing numbers.
- It defines a set of symbols (digits) and a set of rules for how those symbols are combined to represent numerical values.
- Key characteristics of most number systems are: *Base* (or Radix) and *Place Value* (or Digit Position value).

1.1. Base

- This is the total number of unique digits or symbols used in the system.
- For example *Decimal* (Base-10) uses 10 digits (0-9).
- *Binary* (Base-2) uses 2 digits (0-1).
- *Octal* (Base-8) uses 8 digits (0-7).
- *Hexadecimal* (Base-16) uses 16 digits (0-9,A-F).

1.2. Place Value (Digit position value)

- The value of each digit depends on its position within the number.
- Each position represents a power of the base.
- Moving from right to left, the positions correspond to increasing powers of the base ($base^0$, $base^1$, $base^2$, ...etc.).
- Polynomial Formula is as follows (n is the digit position):

$$\text{number} = (\text{digit}_n \times \text{base}^n) + (\text{digit}_{n-1} \times \text{base}^{n-1}) + \ldots + (\text{digit}_1 \times \text{base}^1) + (\text{digit}_0 \times \text{base}^0)$$

- For example, in the decimal base $123 = (1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$.

1.3. Base-5 example

- Our natural number system is base-10 because we have 10 fingers to count.
- Let pretend we have only 5 fingers, and use only them to count.
- We have only 5 digits (0, 1, 2, 3, 4)
- While counting, when we reach the digit 4 we add 1 to the next higher digit (digit at left).

Base-10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Base-5	0	1	2	3	4	10	11	12	13	14	20	21	22	23	24	30	31	32	33

- We can see that 18 in decimals is written 33 in base-5.

- We can verify it by the Polynomial Formula: $18 = (3 \times 5^1) + (3 \times 5^0)$.
- The right mathematical way to write different bases is for the subscripts to denote the base. Ex: $(18)_{10} = (33)_5$

2. Binary Number System (Base-2)

- The binary number system is the foundation of all modern digital computing.
- Unlike the decimal system we use daily, which has ten unique digits, the binary uses only two.
- This simplicity directly reflects the fundamental "on" or "off" states of electrical signals in computer hardware.
- Making it the most natural language for machines.

2.1. Digits 0 and 1

- The binary system, known as Base-2, employs just two distinct symbols: 0 (zero) and 1 (one), referred to as bits.
- A 0 typically represents an "off" or "low voltage" state in electronics.
- A 1 typically represents an "on" or "high voltage" state in electronics.

2.2. Converting Binary-to-Decimal (Polynomial Formula)

- In binary, each digit's position (or place value) corresponds to a power of 2.
- The value of each position increases as you move from right to left.
- Forming the binary Polynomial Formula (n is the digit position):

$$(\text{Number})_{10} = (\text{digit}_n \times 2^n) + (\text{digit}_{n-1} \times 2^{n-1}) + \ldots + (\text{digit}_1 \times 2^1) + (\text{digit}_0 \times 2^0)$$

- For example, following the Polynomial Formula, the number $(110101)_2 = (53)_{10}$

$(1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) =$
$(1 \times 32) + (1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) = 53$

- With this method, it is possible to represent any decimal number.
- This method is the main method to convert a binary value to a decimal value.
- Using the table below, it is possible to simplify the calculation of a binary number by summoning only the power of 2 with a place value of 1.

1	1	0	1	0	1	Number
32	16	8	4	2	1	Power of 2
32	16		4		1	Sum = 53

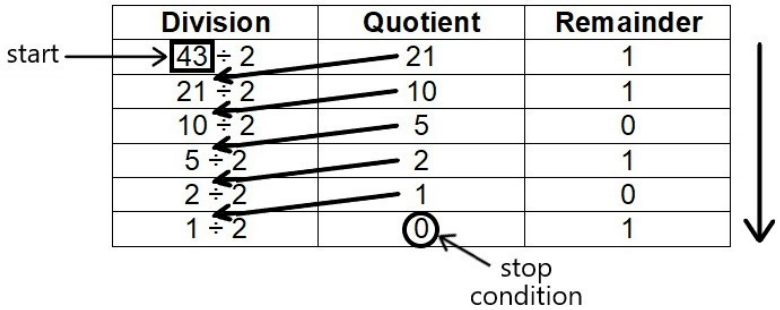
- This is why it is important to memorize the powers of 2:

10	9	8	7	6	5	4	3	2	1	0	Digit position
1024	512	256	128	64	32	16	8	4	2	1	Power of 2

Remark 1: The digit in position 0 in a binary representation is often called the *Least Significant Bit* (or LSB), and the higher (leftest bit) is called the *Most Significant Bit* (MSB).

2.3. Converting Decimal-to-Bianry (Successive Euclidean Division)

- Polynomial Formula was used to convert Binary-to-Decimal.
- The Successive Euclidean Division is a method used in reverse, converting Dicimal-to-Binary.
- The method is based on a successive division by 2 of the decimal number.
- After a division, the quotient of the division is divided again repeatedly.
- The repeated divisions stop when the quotient is equal to 0.
- The 1s and 0s remind of the divisions by 2 form the binary value.
- The reminder of the first division is the LSB of the binary number, and the last reminder is the MSB.
- Let perform an example on the decimal number $(43)_{10}$:



$$(43)_{10} = (101011)_2$$

<u>Remark 2</u>: It is actually common to find the subscript 2 replaced by *b* to represent binary numbers, like $(101)_b$.

3. Octal Number System (Base-8)

- While binary is the native language of computers, it can be quite verbose and difficult for humans to read and write long strings of 0s and 1s.
- The octal number system was historically used as a more compact and human-friendly representation of binary numbers.
- Particularly when dealing with computer architectures that handle data in groups of three bits.

3.1. Digits 0 to 7

- The octal system, or Base-8, uses eight unique digits: 0, 1, 2, 3, 4, 5, 6, 7.
- It does not use the digits 8 or 9.

3.2. Converting Octal-to-Decimal (Polynomial Formula)

- Similar to decimal and binary, each position in an octal number has a specific place value.

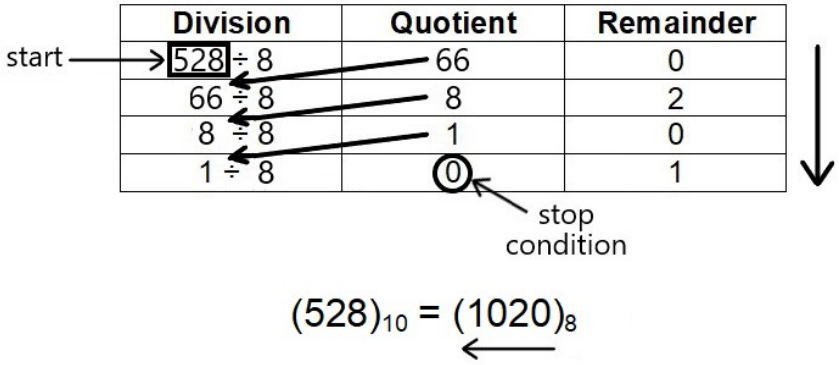
- The place value is a power of its base 8. Moving from right to left, the place values form the octal Polynomial Formula:

$$(\text{Number})_{10} = (\text{digit}_n \times 8^n) + (\text{digit}_{n-1} \times 8^{n-1}) + \ldots + (\text{digit}_1 \times 8^1) + (\text{digit}_0 \times 8^0)$$

- For example, following the Polynomial Formula, the number $(123)_8 = (83)_{10}$
  $(1 \times 8^2) + (2 \times 8^1) + (3 \times 8^0) = (1 \times 64) + (2 \times 8) + (3 \times 1) = 83$

3.3. Converting Decimal-to-Octal (Successive Euclidean Division)

- The same method, Successive Euclidean Division, is used to convert Dicimal-to-Octal.
- Only one change is required in the division, is to use 8 instead of 2.
- Let perform an example on the decimal number $(528)_{10}$:



$$(528)_{10} = (1020)_8$$

3.4. Relationship Octal-Binary

The most significant advantage of octal numbers lies in their direct and simple relationship with binary numbers. This relationship made octal a convenient shorthand in early computing.

- Since $2^3 = 8$, each single octal digit can be perfectly represented by a group of three binary digits (bits).
- We have in the table below Octal-to-3-bit-Binary equivalents:

Octal digit	3-bits equivalent
$(0)_8$	$(000)_2$
$(1)_8$	$(001)_2$
$(2)_8$	$(010)_2$
$(3)_8$	$(011)_2$
$(4)_8$	$(100)_2$
$(5)_8$	$(101)_2$
$(6)_8$	$(110)_2$
$(7)_8$	$(111)_2$

3.4.1. Octal-to-Binary conversion (3-bits equivalence)

- Using the table above, it becomes very easy to convert from octal to binary, it is just a replacement of the octal digit by its binary equivalent in the table.
- Leading zeros can be omitted

- For example: $(223)_8 = (\underline{0}10|010|011)_2 = (101010011)_2$

3.4.2. Binary-to-Octal conversion (3-bits equivalence)

- It is practically the opposite procedure to convert from binary to octal.
- Starting from rightmost digit, replacing each group of 3-bits with its equivalent according to the table.
- Leftmost group may need padding by 0s, if it is not complete.
- For example: $(1010111)_2 = (\underline{00}1|010|111)_2 = (127)_8$

4. Hexadecimal Number System (Base-16)

- The hexadecimal number system is widely used in computing and digital electronics.
- Its primary purpose is to provide a more compact and human-readable representation of binary numbers. Making long binary strings much easier to manage.
- Data is predominantly organized into 8-bit bytes (and multiples thereof). Hexadecimal is particularly well-suited because each hexadecimal digit perfectly represents four binary digits.

4.1. Digits 0-9 and A, B, C, D, E, F

- The hexadecimal system, or Base-16, uses sixteen unique symbols.
- These include the familiar decimal digits 0 through 9.
- Then the first six letters of the alphabet, A, B, C, D, E, F, to represent the decimal values 10, 11, 12, 13, 14, 15 respectively.

4.2. Converting Hexadecimal-to-Decimal (Polynomial Formula)

- Similar to decimal, binary, and octal, each position in a hexadecimal number has a specific place value.
- The place value is a power of its base 16. Moving from right to left, the place values form the Hexadecimal Polynomial Formula:

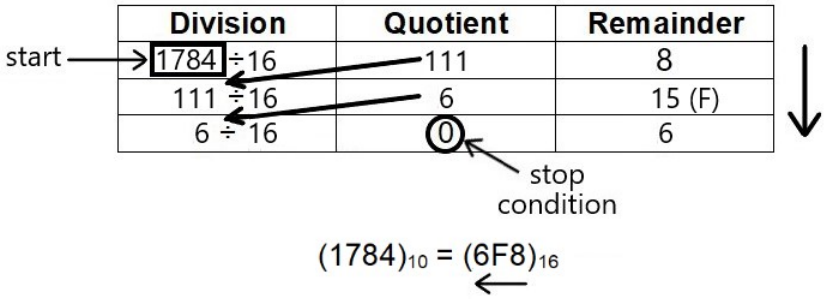
$$(\text{Number})_{10} = (\text{digit}_n \times 16^n) + (\text{digit}_{n-1} \times 16^{n-1}) + \ldots + (\text{digit}_1 \times 16^1) + (\text{digit}_0 \times 16^0)$$

- For example, following the Polynomial Formula, the number $(B57)_{16} = (2903)_{10}$

$$(11 \times 16^2) + (5 \times 16^1) + (7 \times 16^0) = (11 \times 256) + (5 \times 16) + (7 \times 1) = 2903$$

4.3. Converting Hexadecimal-to-Decimal (Successive Euclidean Division)

- The same method, Successive Euclidean Division, is used to convert Dicimal-to-Hexadecimal.
- Only one change is required in the division, is to use 16 instead of 8.
- Let perform an example on the decimal number $(1784)_{10}$:



4.4. Relationship Hexadecimal-Binary

Hexadecimal's primary advantage stems from its compact representation of binary data, particularly in systems where data is handled in 8-bit chunks (bytes) or 16-bit, 32-bit, or 64-bit words. It acts as a crucial bridge, allowing humans to interact with and understand the binary world of computers more effectively.

- Since 2^4 = 16, like octal, each single hexadecimal digit can be perfectly represented by a group of three binary digits (bits).
- We have in the table below Hexadecimal-to-4-bit-Binary equivalents:

Hexadecimal digit	4-bits equivalent
$(0)_{16}$	$(0000)_2$
$(1)_{16}$	$(0001)_2$
$(2)_{16}$	$(0010)_2$
$(3)_{16}$	$(0011)_2$
$(4)_{16}$	$(0100)_2$
$(5)_{16}$	$(0101)_2$
$(6)_{16}$	$(0110)_2$
$(7)_{16}$	$(0111)_2$
$(8)_{16}$	$(1000)_2$
$(9)_{16}$	$(1001)_2$
$(A)_{16}$	$(1010)_2$
$(B)_{16}$	$(1011)_2$
$(C)_{16}$	$(1100)_2$
$(D)_{16}$	$(1101)_2$
$(E)_{16}$	$(1110)_2$
$(F)_{16}$	$(1111)_2$

4.4.1. Hexadecimal-to-Binary conversion (4-bits equivalence)

- Using the table above, it becomes very easy to convert from hexadecimal to binary, it is just a replacement of the hexadecimal digit by its binary equivalent.
- Leading zeros can be omitted
- For example: $(3D)_{16} = (\underline{00}11|1101)_2 = (111101)_2$

4.4.2. Binary-to-Hexadecimal conversion (4-bits equivalence)

- It is practically the opposite procedure to convert from binary to hexadecimal.
- Starting from rightmost digit, replacing each group of 4-bits with its equivalent according to the table.

- Leftmost group may need padding by 0s, if it is not complete.
- For example: $(101010)_2 = (\underline{00}10|1010)_2 = (2A)_{16}$

5. Representation of Fractional Numbers

- Just as decimal numbers can have a fractional part (ex: 0.5), any other number system can also represent values less than one.
- This is achieved by extending the place value system to include negative powers of the base after a fractional point (analogous to a decimal point).
- Like for example in decimals: $0.54 = 5x10^{-1} + 4x10^{-2}$
- A fractional part of a number with any base can be represented by a Fractional Polynomial Formula as follows:

$$\text{fraction} = (\text{digit}_{-1} \times \text{base}^{-1}) + (\text{digit}_{-2} \times \text{base}^{-2}) + \ldots..$$

- That leads to getting a Generalized Polynomial Formula:

$$\text{fractional number} = (\text{digit}_n \times \text{base}^n) + (\text{digit}_{n-1} \times \text{base}^{n-1}) + \ldots + (\text{digit}_1 \times \text{base}^1) + (\text{digit}_0 \times \text{base}^0) + (\text{digit}_{-1} \times \text{base}^{-1}) + (\text{digit}_{-2} \times \text{base}^{-2}) + \ldots..$$

5. Representation of Fractional Numbers

At this point we need to check the possible conversions in fractions between decimal, binary, octal, and hexadecimal.

5.1. Binary-to-Decimal fraction conversion (Fractional Polynomial Formula)

- For the fractional part, multiply each binary digit by its corresponding negative power of 2 (starting with 2^{-1} for the digit immediately after the binary point).
- The value of each position decreases as you move from left to right, after the binary point.
- Forming the binary Fractional Polynomial Formula:

$$(\text{Fraction})_{10} = (\text{digit}_{-1} \times \text{base}^{-1}) + (\text{digit}_{-2} \times \text{base}^{-2}) + \ldots$$

- For example, the number $(0.101)_2 = (0.625)_{10}$
  $(1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) = (1 \times 0.5) + (0 \times 0.25) + (1 \times 0.125) = 0.625$

5.2. Decimal-to-Binary fraction conversion (Successive Multiplication)

- To convert a decimal fraction, the part after the decimal point to binary. A method called Successive Multiplication is applied.
- First step, multiply the decimal fraction by 2.
- The integer part of the result (which will be 0 or 1) becomes the next binary digit after the binary point.
- Discard the integer part and continue multiplying the new fractional part by 2.
- Repeat until the fractional part becomes 0.
- Or you reach a desired number of binary places (as some decimal fractions may

result in non-terminating binary fractions).

- Let perform an example on the decimal number $(0.8125)_{10}$:

Multiplication	Integer part
0.8125 x 2 = 1.625	1
0.625 x 2 = 1.25	1
0.25 x 2 = 0.5	0
0.5 x 2 = 1.0	1

stop condition

$$(0.8125)_{10} = (0.1101)_2$$

5.3. Octal-to-Decimal fraction conversion (Fractional Polynomial Formula)

- Similar to decimal and binary fractions, each position before the point in an octal number has a negative place value.
- The decreasing negative place value is a power of its base 8. Moving from left to right starting from the point.
- The place values form the octal Fractional Polynomial Formula:

$$(\text{Fraction})_{10} = (\text{digit}_{-1} \times 8^{-1}) + (\text{digit}_{-2} \times 8^{-2}) + (\text{digit}_{-3} \times 8^{-3}) + \ldots$$

- For example, following the Fractional Polynomial Formula, $(0.42)_8 = (0.53125)_{10}$
  $(4 \times 8^{-1}) + (2 \times 8^{-2}) = (4 \times 0.125) + (2 \times 0.015625) = 0.53125$

5.4. Decimal-to-Octal fraction conversion (Successive Multiplication)

- The same method, Successive Multiplication, is used to convert Decimal-to-Octal.
- Only one change is required in the multiplication, it is to use 8 instead of 2.
- Let perform an example on the decimal number $(0.15625)_{10}$:

Multiplication	Integer part
0.15625 x 8 = 1.25	1
0.25 x 8 = 2.0	2

stop condition

$$(0.15625)_{10} = (0.12)_2$$

5.5. Hexadecimal-to-Decimal conversion (Fractional Polynomial Formula)

- Similar to octal and binary fractions, each position before the point in an hexadecimal number has a negative place value.
- The decreasing negative place value is a power of its base 16. Moving from left to right starting from the point.
- The place values form the hexadecimal Fractional Polynomial Formula:

$$(\text{Fraction})_{10} = (\text{digit}_{-1} \times 16^{1}) + (\text{digit}_{-2} \times 16^{-2}) + (\text{digit}_{-3} \times 16^{-3}) + \ldots$$

- For example, following the Fractional Polynomial Formula $(0.4C)_{16} = (0.296875)_{10}$
  $(4 \times 16^{-1}) + (12(C) \times 16^{-2}) = (4 \times 0.0625) + (12 \times 0.00390625) = 0.296875$

5.6. Decimal-to-Hexadecimal fraction conversion (Successive Multiplication)

- The same method, Successive Multiplication, is used to convert Dicimal-to-Hexadecimal.
- Only one change is required in the multiplication, it is to use 16 instead of 8.
- Let perform an example on the decimal number $(0.546875)_{10}$:

Multiplication	Integer part
0.546875 x16= 8.75	8
0.75 x16=12.0	12(C)

stop condition

$$(0.546875)_{10} = (0.C8)_{2}$$

5.8. Octal-to-Binary fraction conversion (3-bits equivalent)

- The conversion is very easy and similar to the non-fractional part of Octal-to-Binary conversion, except it is done in the opposite way to the point.
- It is just a replacement of the octal digit by its binary equivalent.
- Tailing zeros can be omitted.
- For example: $(0.34)_8 = (0.011|1\underline{00})_2 = (0.0111)_2$

5.7. Binary-to-Octal fraction conversion (3-bits equivalent)

- Using the table of equivalence, it becomes very easy to convert from Binary-to-Octal.
- First, group binary digits into sets of three moving from left to right from the binary point.
- And padding with trailing zeros if necessary.
- Convert each 3-bit group to its octal equivalent from the table.
- For example: $(0.1011)_2 = (0.101|1\underline{00})_2 = (0.54)_8$

5.9. Hexadecimal-to-Binary fraction conversion (4-bits equivalent)

- Similar to the Octal-to-Binary fraction conversion, the conversion Hexadecimal-to-Binary is very easy.
- It is just a replacement of the hexadecimal digit by its binary equivalent.
- Tailing zeros can be omitted.

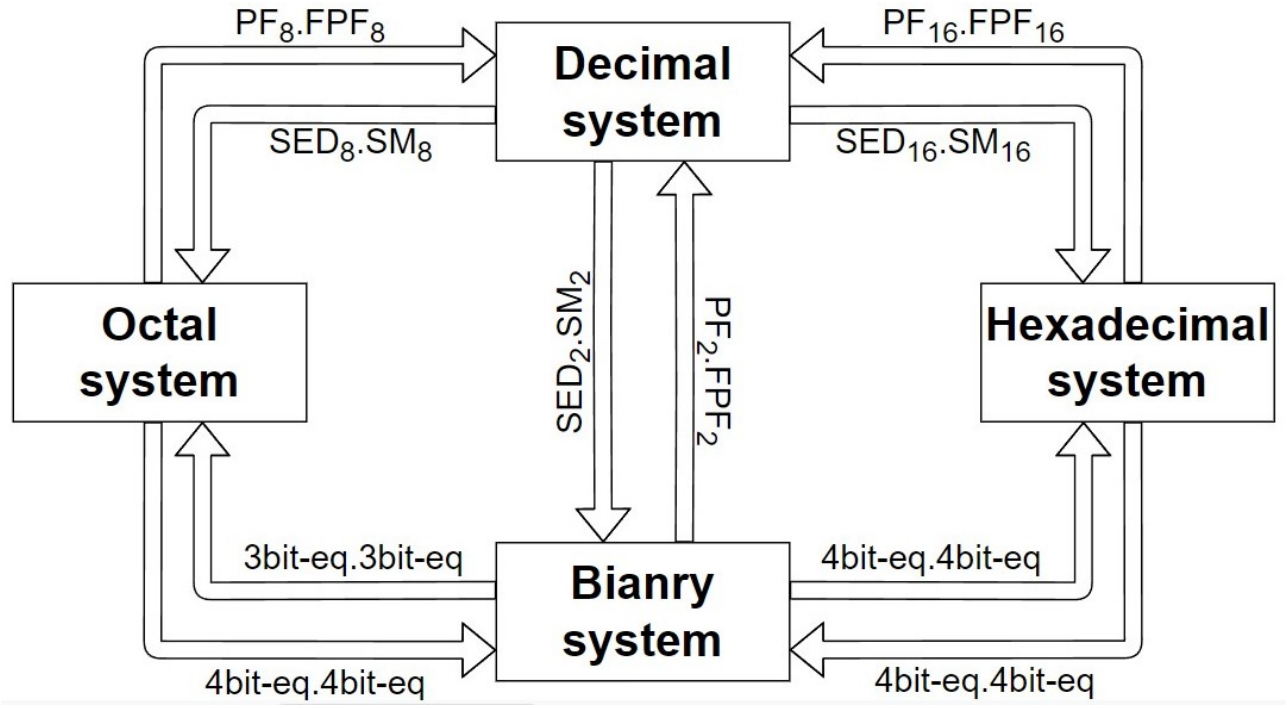
- For example: $(0.2C)_{16} = (0.0010|11\underline{00})_2 = (0.001011)_2$

5.10. Binary-to-Hexadecimal fraction conversion (4-bits equivalent)

- The same, it becomes very easy to convert from Binary-to-Hexadecimal.
- First, group binary digits into sets of four moving from left to right from the binary point.
- And padding with trailing zeros if necessary.
- Convert each 4-bit group to its hexadecimal equivalent from the table.
- For example: $(0.1010101)_2 = (0.1010|101\underline{0})_2 = (0.BB)_{16}$

6. A summary schematic of all conversions

A high number of methods are used to convert from different bases, decimal, binary, octal, hexadecimal. Below, a diagram is presented to summarize all the methods needed for each conversion.



The table below gives the diagram legends :

Legends	Description
SED_n	Successive Euclidean Division by the base *n*
SM_n	Successive Multiplication by the base *n*
PF_n	Polynomial Formula with base *n*
FPF_n	Fractional Polynomial Formula with base *n*
3bit-eq	3-bits equivalent digit
4bit-eq	4-bits equivalent digit

Remark 3: You can observe in the diagram there is no direct method to convert from Octal-to-Hexadecimal or the opposite. The best way to do it is through a 2-step operation, going by the binary system. Going through decimal system is more complicated.

7. Arithmetic operations

The well-known arithmetic operations addition, subtraction, multiplication, and division. Are executed the same way in decimal system base-10 as any other system, including binary, octal, and hexadecimal. The difference is just to be careful of the respect of the base executed in. In this section, the focus is on binary operations.

7.1. Addition

- Addition like any other operation works the same as in the decimal base-10.
- Only one rule should be respected to guarantee a correct addition in any base. The calculation of the *carry*.
- While adding 2 digits in the same column, the result and the carry should be added with respect to the base.
- Example in the binary base system:

$$\begin{array}{rl} & {\scriptstyle 1\ 1} \\ & 1\,0\,1\,1 \leftarrow 11 \\ + & 0\,0\,1\,1 \leftarrow 3 \\ \hline = & 1\,1\,1\,0 \leftarrow 14 \end{array}$$

7.2. Subtraction

- Similar to addition, subtraction, in any other base, works the same as in decimal system base-10.
- One rule should be respected to guarantee a correct subtraction. The calculation of the *borrow*.
- While borrowing 1 from the next column, the value of the new digit to subtract should be interpreted with respect to the actual base system.
- Example in the binary base syste:

$$\begin{array}{rl} & 1_1 0_1 0_1 0 \leftarrow 8 \\ - & 0\,0\,1\,1 \leftarrow 3 \\ & {\scriptstyle 1\ 1\ 1} \\ \hline = & 0\,1\,0\,1 \leftarrow 5 \end{array}$$

<u>Remark 4</u>: It is possible to get negative numbers. They are represented like negative numbers in a decimal system, using the dash (-). Ex: $(10)_2 - (11)_2 = (-1)_2$

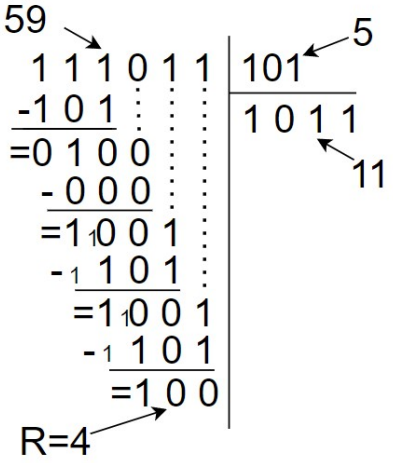
7.3. Multiplication

- Multiplication also works the same as in the decimal base-10 in any base.
- The rule that should be respected to guarantee a correct multiplication is the correct way to calculate the carry.
- The addition over many lines implies respect for the base digits while calculating the result.
- Example in the binary base system:

```
      1 0 1 0 ←— 10
    x 1 0 1 1 ←— 11
    ---------
    = 1 0 1 0
+     1 0 1 0 ·
+   0 0 0 0 · ·
+ 1 0 1 0 · · ·
---------------
= 1 1 0 1 1 1 0 ←— 110
```

7.4. Division

- Division also works like other operations, and the same as in the decimal base-10 in other bases.
- The rule that should be respected to guarantee a correct multiplication is the correct way to calculate the carry.
- The addition over many lines implies respect for the base digits while calculating the result.
- Example in the binary base system:



Remark 5: Even though all arithmetic operations explained above are applicable for any base, the focus was around the binary system. Because these binary operations have to be implemented inside the ALU in the next course (SM2).

Remark 6: The fractional numbers related to arithmetic operations were not discussed, but the same rules used in decimal fractional numbers are also applicable in the same way to any other base.