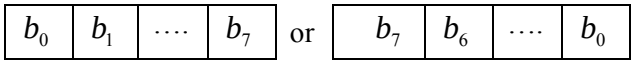
Chapter 2

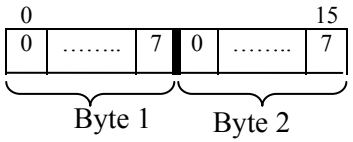
Forms of Information Representation in Digital Computers.

§1. General Considerations

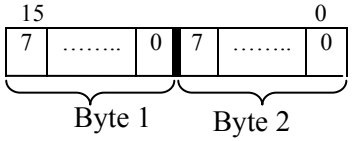
- There are distinguished the following two fundamental modes of information representation: (1.1)
  a) internal;
  b) external;
- Internal representations refers to the methods and means of representation of information in internal devices of the digital computer: gates, bistables, registers, memory, etc. (1.2)
- External representation refers to the methods and means of representation in peripheral and terminal units (usually magnetic / optical media). (1.3)
- In what follows it is presented extensively the internal representation based on use of the binary digits (bits).
- The following concepts are related to a sequence of bits
  - **Byte**: a group of 8 bits (1.4)



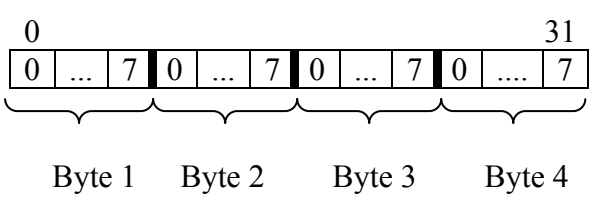
  - **Half – Word**: a group of 2 Bytes (1.5)



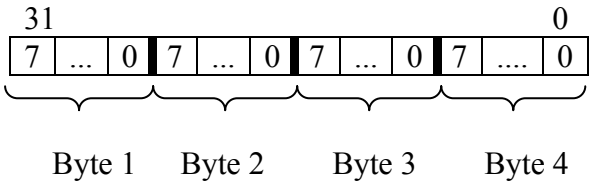
or



  - **Word**: a group 4 Bytes

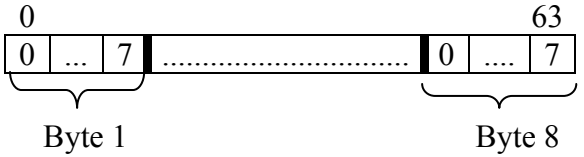


or

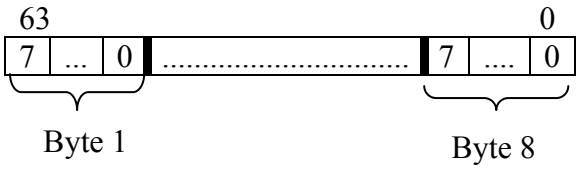


(1.6)

  - **Double Word**: a group of 8 Bytes



or



(1.7)

- With regard to the internal representation the following two kinds are identified: (1.8)

a) **Natural** representation corresponding to fixed point representation, designated **FXP**

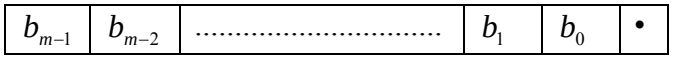
b) **Normal** representation corresponding to floating point representation, designated **FLP**

§2. Fixed-Point Representation of Numbers

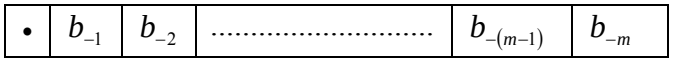
§2.1. Introductory concepts

- Fixed-Point representation of numbers assumes that in all number representations a fixed position of the binary point is adopted. (2.1.1)
  In case of the extreme positions of the binary point there are defined:

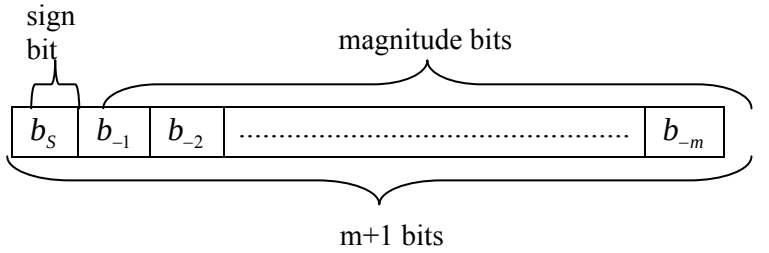
1) *Integer Numbers* → binary point positioned at the right. (2.1.2)



2) *Fractional Numbers* → binary point positioned at the left. (2.1.3)



- For fractional numbers the binary point has a double role: (2.1.3)
  1) to separate the integer part from the fractional part
  2) to separate the sign bit from the magnitude

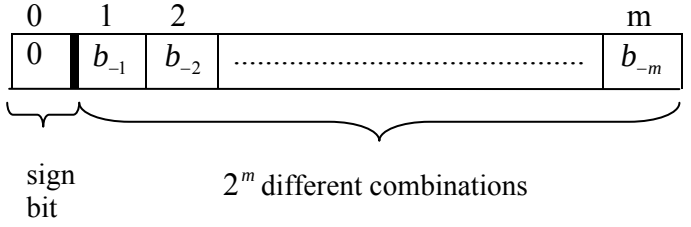


§2.2. Analysis of the representation ranges for positive fractional numbers

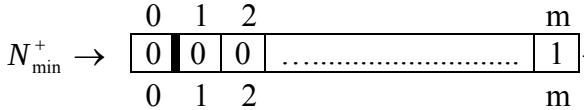
- Assuming that the module of N is: (2.2.1)

$$|N| = \sum_{-m}^{-1} b_i \cdot 2^i$$

- The positive fractional numbers are represented as follows: (2.2.2)



- Hence, 2^m positive numbers can be defined in the considered format (1, m) (2.2.3)
- Extreme values:



(2.2.4) (2.2.5)

- Hence, the range of positive fractional numbers is: (2.2.6)

$$N^+ \in [2^{-m}, (1-2^{-m})]$$

§2.3. Analysis of the representation ranges for negative fractional numbers

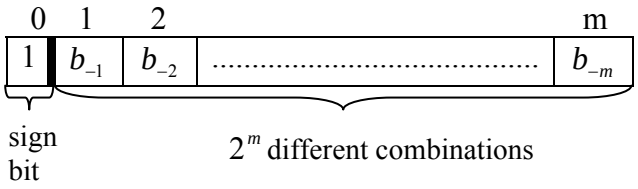
- The analysis is done for the three codes used for negative number representation : sign-magnitude, two's complement code, one's complement code. (2.3.1)

A. Sign-Magnitude Code (2.3.2)

- Assuming that the module of N is:

$$|N| = \sum_{-m}^{-1} b_i \cdot 2^i$$

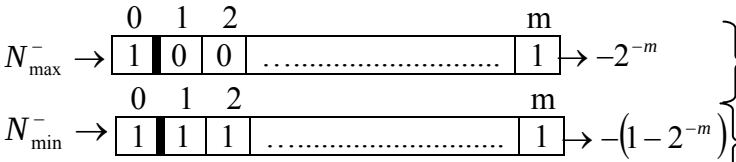
- The negative fractional number $|N|$ is represented as follows*:*



(2.3.3)

- Hence, 2^m negative numbers can be defined in the considered format (1, m) (2.3.4)

- The extreme values:



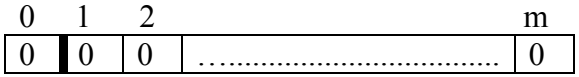
(2.3.5) (2.3.6)

Hence, the range of negative numbers is:

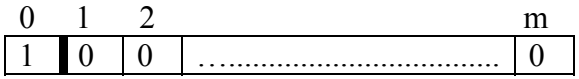
- $N^- \in [-2^{-m}, -(1-2^{-m})]$ (2.3.7)

- Two representations for 0 :

➢ Positive zero



➢ Negative zero



(2.3.8)

- Overflow (**OVF**)

The overflow condition arises whenever an attempt of representing a number whose module is greater than $1-2^{-m}$ occurs.

Condition: $|N| > 1-2^{-m}$ (2.3.9)

➢ **OVF**$^+$ positive if $N > 1-2^{-m}$

➢ **OVF**$^-$ negative if $N < -(1-2^{-m})$

OVF → singularity case (2.3.10)

- Underflow (**UNF**)

The underflow condition arises whenever an attempt of representing a number whose module is less than 2^{-m} occurs.

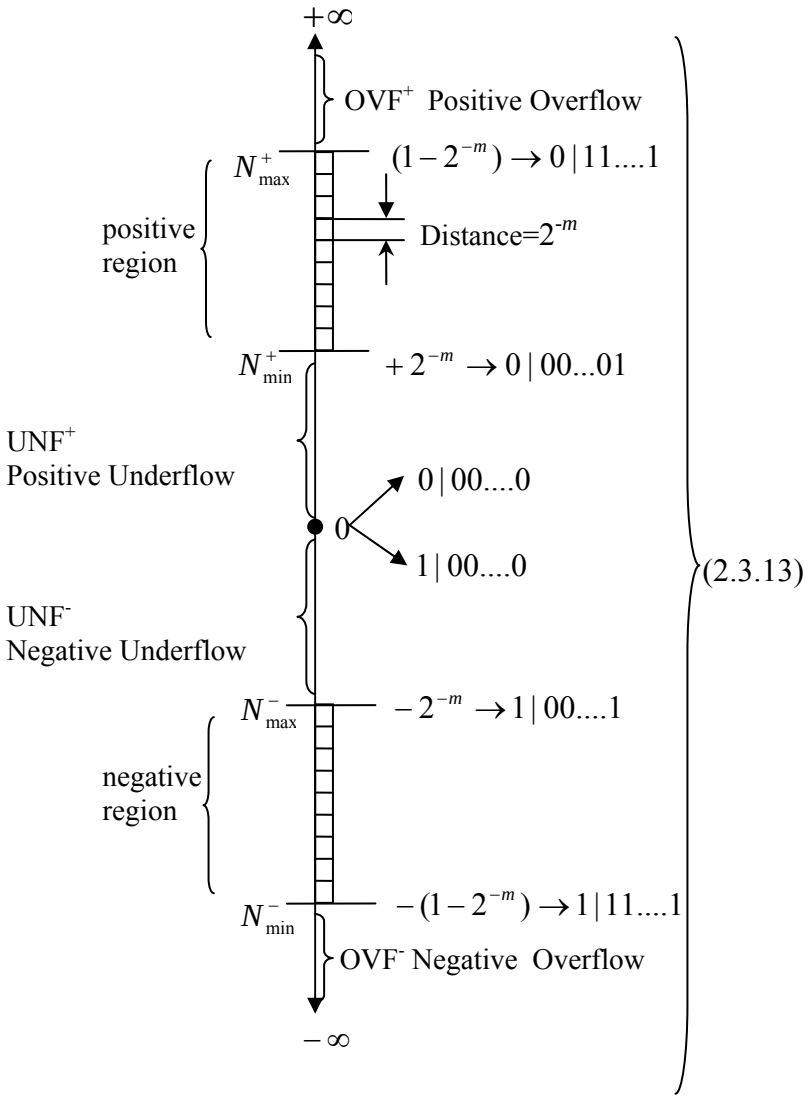
Condition : $|N| < 2^{-m}$ (2.3.11)

➢ **UNF**$^+$ positive if $N < 2^{-m}$

➢ **UNF**$^-$ negative if $N > -2^{-m}$

UNF may not be treated as error (2.3.12)

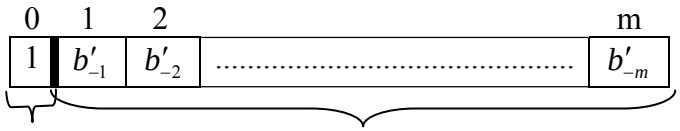
- Representation on the real axis :



(2.3.13)

B. Two's Complement Code

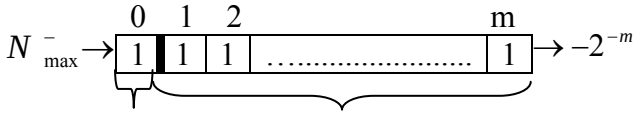
- The discussion is restricted to the first variant (2.3.14)
- General form for -|*N*|:



where: $\sum_{-m}^{-1} b'_i \cdot 2^i = \sum_{-m}^{-1} \overline{b_{-i}} \cdot 2^i + 2^{-m}$ (two's complement representation)

(2.3.15)

- The extreme values:

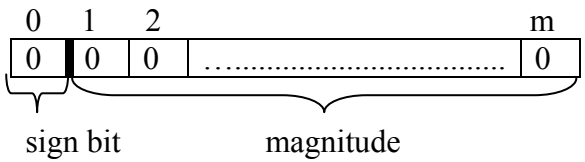


(2.3.16)

(2.3.17)

- Hence
  $N \in [-2^{-m}, -(1-2^{-m})]$ (2.3.18)

- A single representation for 0 :

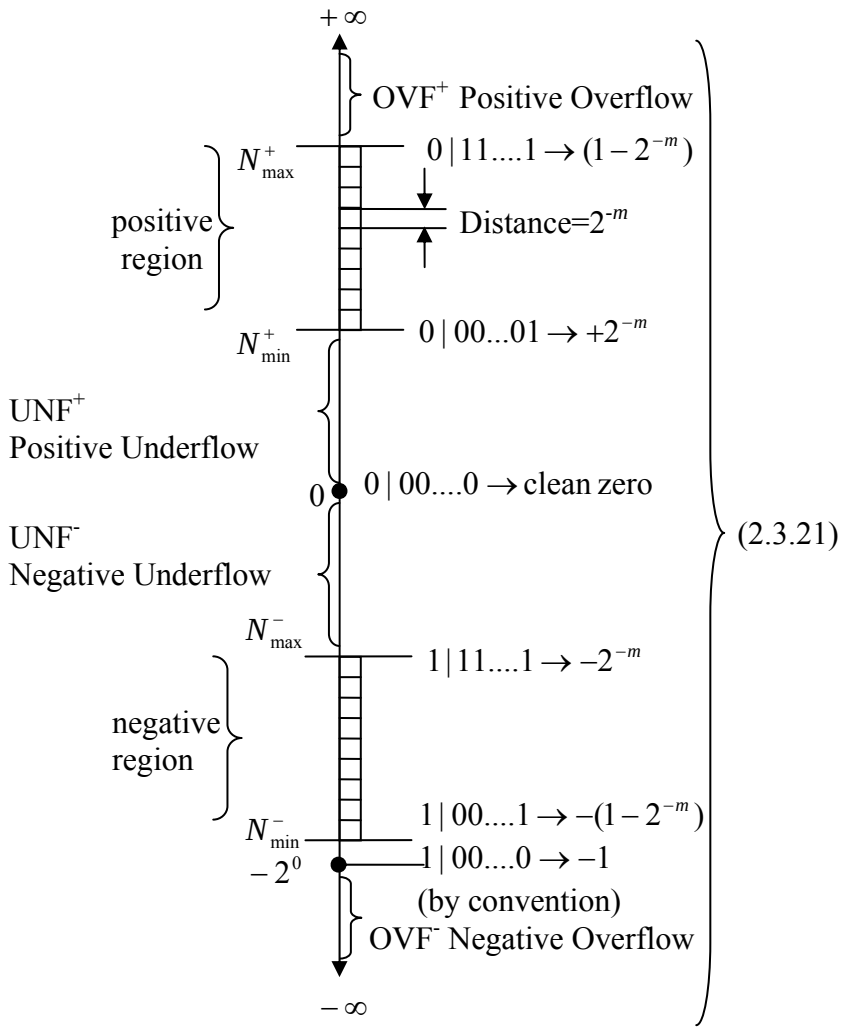


(2.3.19)

- The dirty 0, corresponding to minus 0, that is $1\,|\,\underbrace{000...0}_{m}$, is used *by convention* for representation of the particular value -2^0

(2.3.20)

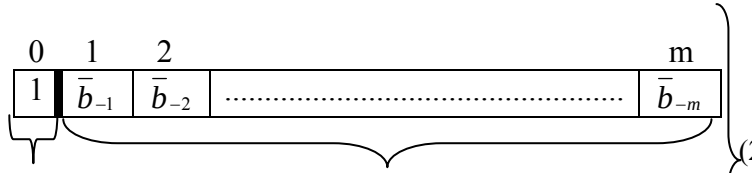
- Representation on real axis:



(2.3.21)

C. One's Complement Code

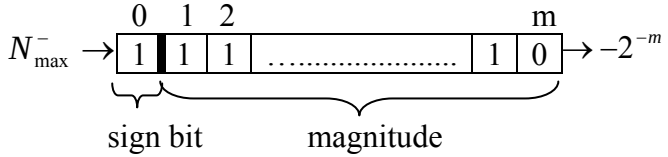
- The discussion is restricted to the first variant (2.3.22)
- General form for -|*N*|:



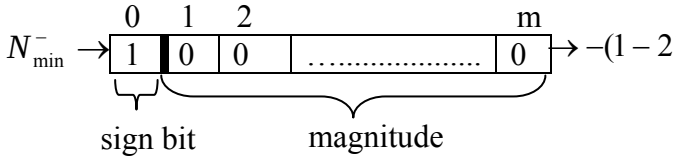
(2.3.23)

where $\bar{b}_i = 1 - b_i$

- The extreme values:

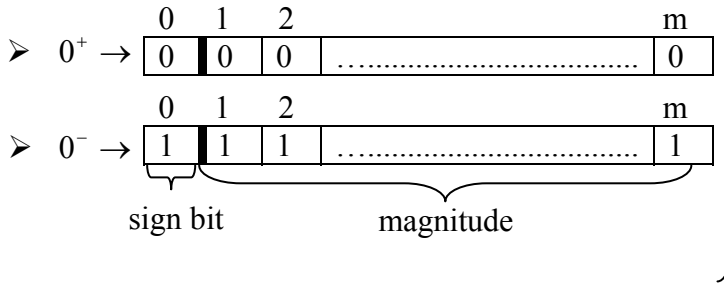


(2.3.24)



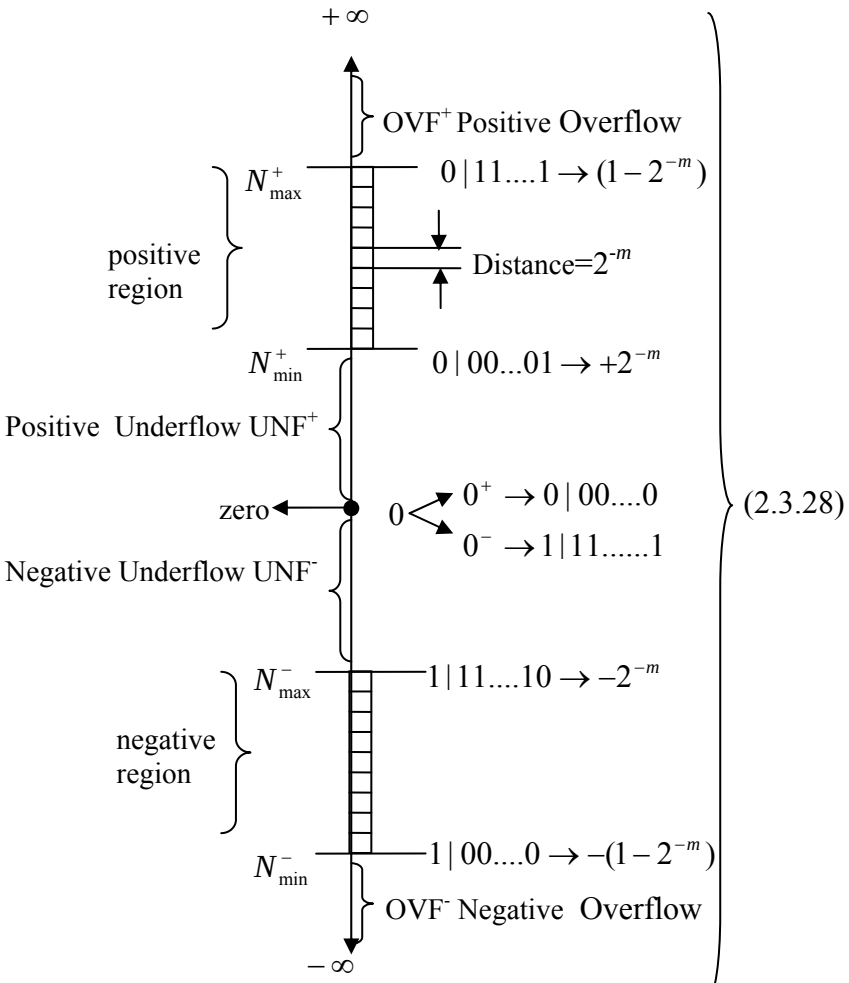
(2.3.25)

- Hence, $N \in [-2^{-m}, -(1-2^{-m})]$ (2.3.26)
- Two representations for 0 :



(2.3.27)

- Representation on real axis :



(2.3.28)

§3. Floating point representation of numbers

§3.1. General considerations

- FXP notations are convenient for representing small numbers with bounded orders of magnitude (1.1)

For instance, if

n=32 bits, then $N = \pm\left(2^{31}-1\right) \rightarrow \sim \pm 10^{11}$ (1.2)

This range is inadequate for engineering and scientific applications

- FLP uses a two part representation:

$$N_{FLP} = (m, e) \quad (1.3)$$

where m is the **mantissa** and e is the exponent of FLP representation.

- Any number N can be represented in the following form

$$N = m \cdot r^e \quad (1.4)$$

where: (1.5)

- e=**exponent** (integer)
- m=**mantissa** (fraction)
- m, e are signed fixed point numbers
- r=radix (self-implied)

- *Example*:

$$r = 10$$
$$N = 9.25$$
$$N = 10^{+1} \cdot 0.925 \quad (1.6)$$
$$m = 0.925$$
$$e = +1$$

- For the particular case r=2

$$N_2 = m_2 \cdot 2^{e_2} \quad (1.7)$$

- *Example:*

$$N_2 = 0.0101$$
$$N_2 = 2^{-1} \cdot 0.101 \quad (1.8)$$
$$m = 0.101$$
$$e = -1$$

- Main advantages of FLP notation (1.9)
  - drastic enlarging of the range of representation
  - constant relative error of representation
- Main drawbacks of FLP notation (1.10)
  - cost of implementation
  - arithmetic algorithms more complex

§3.2. FLP formats

- **Normalized** FLP representation

$$|m| \in \left[\frac{1}{r}, 1\right)$$

$$\frac{1}{r} \le |m| < 1 \quad (2.1)$$

- for r=2

$$|m| \in \left[\frac{1}{2}, 1\right)$$

$$\frac{1}{2} \le |m| < 1 \quad (2.2)$$

the most significant bit of the mantissa is always 1.

- *Example*:

$$N_2 = 0.000101$$
$$N_2 = 0.101 \cdot 2^{-3} \quad (2.3)$$

Normalized mantissa: .101
Exponent: -3 represented as integer FXP
Both mantissa and exponent are represented in FXP notation. (2.3)

- **Un-normalized** FLP representation

$$|m| < \frac{1}{r} \quad (2.4)$$

- for r=2:

$$|m_2| < \frac{1}{2} \quad (2.5)$$

The most significant bits of the mantissa are zero

- *Example*:

$N_2 = 0.000101$

$N_2' = 0.00101 \cdot 2^{-1}$, where m' = .00101 and e' = -1

$N_2'' = 0.0101 \cdot 2^{-2}$, where m'' = .0101 and e' = -2 (2.6)

- In computer design it is preferred normalized FLP Notation. where the most significant position of the mantissa contains a non-zero digit, which is *unique* for the number N. (2.7)

- **Normalizing operation** consists in shifting the mantissa and adjusting the exponent until normalization condition is derived. (2.8)

- **Rules** for normalization operation:

a) for one left shift of mantissa decrement the exponent
b) for one right shift of mantissa increment the exponent (2.9)

- *Example 1:*

Given the un-normalized FLP binary number

$$N_2 = 2^4 \cdot 0.001101 \quad (2.10)$$

After normalizing operation

$$N_2 = 2^2 \cdot 0.1101 \quad (2.11)$$

Thus, m = .1101, e = +2

- *Example 2:*

Given the un-normalized FLP binary number

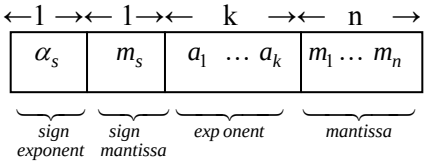
$$N_2 = 2^{-1} \cdot 1.01101 \quad (2.12)$$

After normalizing operation:

$$N_2 = 2^0 \cdot 0.101101 \quad (2.13)$$

Thus, m = .101101, e = 0

- **General format of FLP notation**



(2.14)

where: (2.15)

- α_s = sign bit of the exponent
- m_s = sign bit of the mantissa
- k = number of bits for exponent representation
- n = number of bits for mantissa representation
- a_i = bit of the exponent
- m_i = bit of the mantissa

- The exponent is either **unbiased** or **biased**; the biased exponent assumes that always the exponent is positive, not requiring the sign bit α_s (2.16)

• **Biased** exponent:

$C = E + T$, such that $C \geq 0$

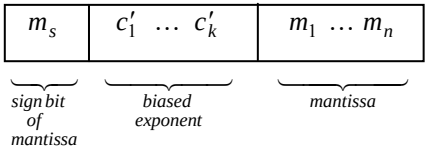
where

$T = \frac{r^k}{2}$, the biasing constant

for $r = 2$, $T = 2^{k-1}$

(2.17)

• Modified format of FLP notation



(2.18)

• *Example 1:*

$r = 10$

$k = 2$

$T = \frac{10^k}{2} = \frac{100}{2} = 50$

Biased exponent in the range: $[0 \ , 99]$

$C_{\min} = 0$

$C_{\max} = 99$

since $T = 50$ it results that:

$C = E + 50$

$C_{\min} = E_{\min} + 50 \rightarrow E_{\min} = C_{\min} - 50 = -50$

$C_{\max} = E_{\max} + 50 \rightarrow E_{\max} = C_{\max} - 50 = 49$

Thus, the actual range of exponent is

$E \in [-50 \ , +49]$

(2.19)

• *Example 2:*

$r = 2$

$k = 4$

$T = \frac{2^4}{2} = 8$

Biased exponent in the range: $[0 \ , 15]$

$C_{\min} = 0000 \rightarrow 0$

$C_{\max} = 1111 \rightarrow 15$

since $T = 8$ it results that:

$C = E + 8$

$C_{\min} = E_{\min} + 8 \rightarrow E_{\min} = C_{\min} - 8 = \ 0 - 8 = -8$

$C_{\max} = E_{\max} + 8 \rightarrow E_{\max} = C_{\max} - 8 = 15 - 8 = 7$

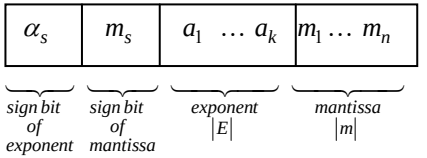
Thus, the actual range of exponent is

$E \in [-8 \ , +7]$

(2.20)

§3.3. Representation ranges for binary FLP numbers

• Standard FLP format



(3.1)

• Normalized FLP numbers case:

$\frac{1}{2} \leq |m| < 1$ (3.2)

$|m|_{\min_n} = 2^{-1}$ (3.3)

$$|m|_{\max_n} = 1 - 2^{-n} \quad (3.4)$$

$$|E|_{\min_n} = 0 \quad (3.5)$$

$$|E|_{\max_n} = 2^k - 1 \quad (3.6)$$

$$E_{\max_n} = +(2^k - 1) \quad (3.7)$$

$$E_{\min_n} = -(2^k - 1) \quad (3.8)$$

Hence,

$$N^+_{\max_n} = 2^{E_{\max_n}} \cdot m_{\max_n} = 2^{2^k - 1} \cdot \left(1 - 2^{-n}\right) \quad (3.9)$$

$$N^+_{\min_n} = 2^{E_{\min_n}} \cdot m_{\min_n} = 2^{-\left(2^k - 1\right)} \cdot 2^{-1} \quad (3.10)$$

$$N^-_{\max_n} = -2^{E_{\min_n}} \cdot m_{\min_n} = -2^{-\left(2^k - 1\right)} \cdot 2^{-1} \quad (3.11)$$

$$N^-_{\min_n} = -2^{E_{\max_n}} \cdot m_{\max_n} = -2^{2^k - 1} \cdot \left(1 - 2^{-n}\right) \quad (3.12)$$

- Un-normalized FLP numbers case:

It is considered the limit case for mantissa, where:

$$|m|_{\min_u} = 2^{-n} \quad (3.13)$$

The other conditions remain unchanged:

$$|m|_{\max_u} = 1 - 2^{-n} \quad (3.14)$$

$$E_{\max_n} = +(2^k - 1)$$

$$E_{\min_n} = -(2^k - 1)$$

Then:

- $N^+_{\min_u} = +2^{-\left(2^k - 1\right)} \cdot 2^{-n}$ (3.15)
- $N^-_{\max_u} = -2^{-\left(2^k - 1\right)} \cdot 2^{-n}$ (3.16)
- $N^+_{\max_u} = +2^{\left(2^k - 1\right)} \cdot (1 - 2^{-n})$ (3.17)
- $N^-_{\min_u} = -2^{\left(2^k - 1\right)} \cdot (1 - 2^{-n})$ (3.18)

- *Observations*:
  By comparing the above values the following obvious relations are observed:

$$N^+_{\min_u} < N^+_{\min_n} \quad (3.19)$$

$$N^-_{\max_u} > N^-_{\max_n} \text{ , or } \left|N^-_{\max_u}\right| < \left|N^-_{\max_n}\right| \quad (3.20)$$

$$N^+_{\max_u} = N^+_{\max_n} \quad (3.21)$$

$$N^-_{\min_u} = N^-_{\min_n} \quad (3.22)$$

- **FLP overflow** (**OVF**):
  Any attempt to represent a number that is greater then the greatest representable number is called **overflow**
  General conditions:

$$|N| > 2^{2^k - 1}\left(1 - 2^{-n}\right) \quad (3.23)$$

  - Positive overflow ($\mathbf{OVF}^+$)

$$N > 2^{2^k - 1}\left(1 - 2^{-n}\right) \quad (3.24)$$

  - Negative overflow ($\mathbf{OVF}^-$)

$$N < -2^{2^k - 1}\left(1 - 2^{-n}\right) \quad (3.25)$$

- **FLP underflow** (**UNF**)
  Any attempt to represent a number that is smaller then the smallest representable number is called **underflow** (3.26)
  General conditions:

  normalized: $|N| < 2^{-\left(2^k - 1\right)} \cdot 2^{-1}$

  un-normalized: $|N| < 2^{-\left(2^k - 1\right)} \cdot 2^{-n}$ (3.27)

  - Positive underflow ($\mathbf{UNF}^+$)

Conditions:

normalized: $N < +2^{-\left(2^k - 1\right)} \cdot 2^{-1}$ (3.28)

un-normalized: $N < +2^{-\left(2^k-1\right)} \cdot 2^{-n}$ (3.29)

➢ Negative underflow (**UNF** $^-$)

Conditions:

normalized: $N > -2^{-\left(2^k-1\right)} \cdot 2^{-1}$ (3.30)

un-normalized: $N > -2^{-\left(2^k-1\right)} \cdot 2^{-n}$ (3.31)

- **Zero** representation:

  a) Un-normalized 0

  b) Normalized $\begin{cases} 0^+ \rightarrow N^+_{\min} \\ 0^- \rightarrow N^-_{\max} \end{cases}$ (3.32)

- FLP **distance:**

  Assuming that mantissa is on *n* bits, there are represented two consecutive numbers N_1 and N_2, for the exponent *e*:

  $N_1 = 2^e \cdot m$

  $N_2 = 2^e \cdot \left(m + 2^{-n}\right)$

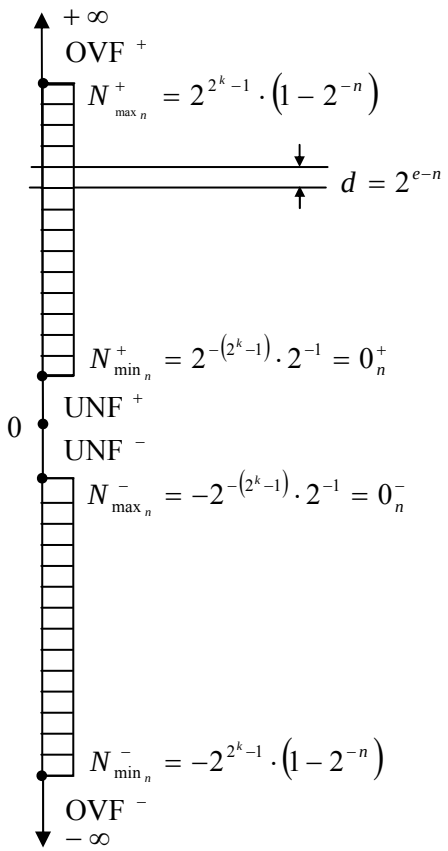
  Then, the distance is given by the difference:

  $d_{FLP} = N_2 - N_1 = 2^e\left(m + 2^{-n}\right) - 2^e \cdot m = 2^{e-n}$ (3.33)

  Thus, the distance *is not constant* as in case of FXP representation, because it depends on the current value of *e*. The greater is *e*, the greater is *d* and vice-versa. (3.34)

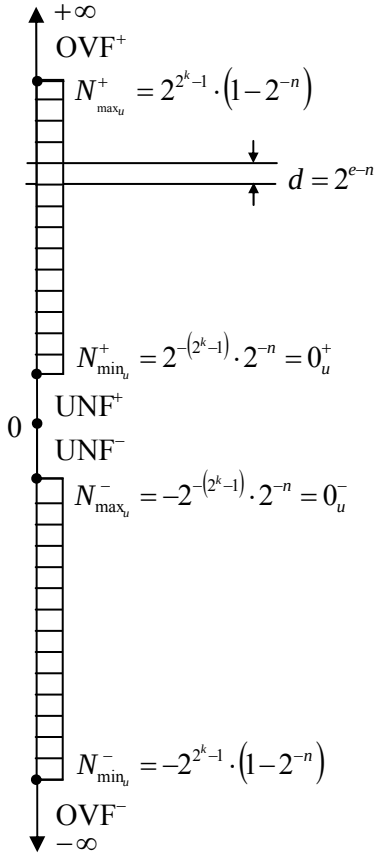
- **Representation of ranges for FLP numbers on real axis:**

a) Normalized numbers



(3.34)

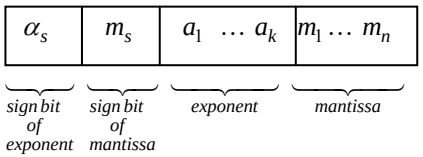
b) Un-normalized numbers



(3.35)

§3.4. Total number of expressible numbers

- Standard FLP format



(4.1)

- Number of different exponents:
  k binary digits, hence
  $|E| \in [0,(2^k-1)] \longrightarrow 2^k$ values
  $E^{+} \in [0,(2^k-1)] \longrightarrow 2^k$ values
  $E^{-} \in [-(2^k-1),0] \longrightarrow 2^k$ values

(4.2)

Total:
$2\times 2^k - 1$ (twicely taken 0)

(4.3)

- Number of different mantissas:
  - n binary digits
  - for the case of normalized FLP numbers
    $|m| \in [2^{-1},(1-2^{-n})] \longrightarrow 2^{n-1}$ values

(4.4)

mantissa | m_{-1} | m_{-2} | …… | m_{-n}

$|m|_{\min}$ | 1 | 0 | …… | 0 | $\longrightarrow 2^{-1}$

$|m|_{\max}$ | 1 | 1 | …… | 1 | $\longrightarrow (1-2^{-n})$

(4.5)

  - ranges for positive and negative mantissas:
    $m^{+} \in [2^{-1},1-2^{-n}] \longrightarrow 2^{n-1}$ values
    $m^{-} \in [-(1-2^{-n}),-2^{-1}] \longrightarrow 2^{n-1}$ values

(4.6)

Total:
$2\times 2^{n-1} = 2^n$

(4.7)

- *Conclusion*: total number of expressible FLP numbers is
  $N = (2^{k+1}-1)\times 2^n$

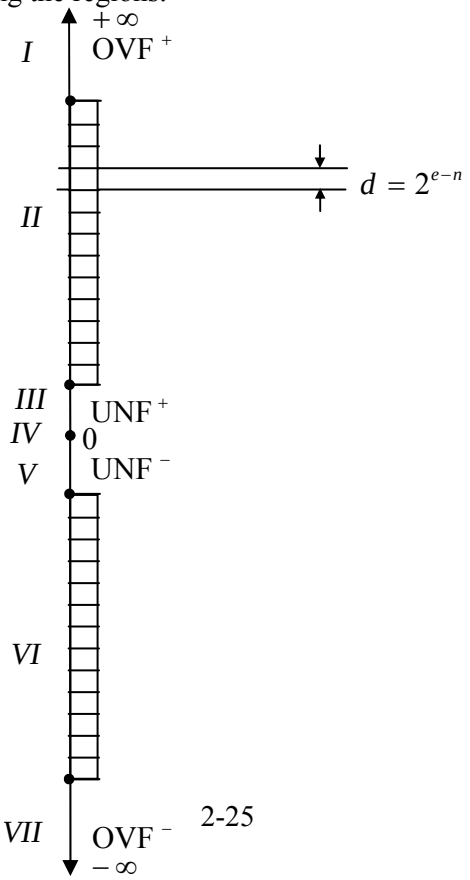
(4.8)

§3.5. FLP representation and Real Number System

- The FLP numbers can be used to *model the real number system* of mathematics, although there are important differences
- The **finite nature of the number representation**, which is unavoidable
- If a number cannot be expressed in the number representation set being used then it will be used the nearest node that can be expressed; this operation is known as ROUNDING operation

(5.1)

Highlighting the regions:



(5.2)

- The following differences are mentioned:
  1) Expressible numbers appear **only** in regions II, IV, VI of real axis (5.3)
  2) **Distance** between two consecutive nodes
     $d_{FLP} = 2^{e-n}$
     Thereby *d* is not constant throughout regions II and VI (5.4)
     Real Numbers are forming Continuum, that is
     $x, y \longrightarrow Z = \frac{x+y}{2}$ (5.5)
  3) Even in regions II,IV,VI there exists a **finite number of nodes**, thereby the DENSITY is finite (5.6)

- Control over regions II and VI is given by the following rules:
  1) By increasing the number of bits at exponent it results the *extension* of regions II and VI and, consequently, the *range of representation* is enlarged (5.7)
  2) By increasing the number of bits at mantissa it results an increase of the number of nodes in regions II and VI, improving the *accuracy of representation*. (5.8)

- *Conclusion:* to comply to the needed accuracy it will be adopted one of different formats of FLP representation: Simple, Double, Extended Precision (SP,DP,EP). (5.9)

§3.6. Example of FLP representation

- Given the decimal integer $N_{10} = 433$
  Assumed FLP format: 1+7+16
  Biased exponent (6.1)

$$C = E + 64$$

- Solution:
  Conversion into radix 2 representation
  $433_{10} = 110110001_2$ (6.2)

- 1) **Normalized representation**
  Fractional notation of *N*:
  $110110001 = 0.110110001 \times 2^9$
  Biased exponent
  $E = 9 \rightarrow C = 9 + 64 = 73 \rightarrow 1001001$
  Required representation:

$$\underbrace{0}_{\text{sign}} \quad \underbrace{1001001}_{\substack{\text{Biased} \\ \text{exponent}}} . \underbrace{1101100010000000}_{\text{mantissa}}$$

(6.3)

- Verification:

$$N = \left(2^{-1} + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-9}\right) \times 2^{73-64} =$$
$$= \left(2^{-1} + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-9}\right) \times 2^{9} =$$
$$= 2^8 + 2^7 + 2^5 + 2^4 + 2^0 =$$
$$= 256 + 128 + 32 + 16 + 1 =$$
$$= 433$$

(6.4)

- 2) **Un-normalized representation**
  Fractional notation of *N*:

$$110110001 = 0.0000\ 0001\ 1011\ 0001 \times 2^{16}$$

  Biased exponent

$$E = 16 \rightarrow C = 64 + 16 = 80 \rightarrow 1010000$$

  Required representation:

(6.5)

$$\underbrace{0}_{\text{sign}} \quad \underbrace{1010000}_{\substack{\text{Biased} \\ \text{exponent}}} . \underbrace{0000\ 0001\ 1011\ 0001}_{\text{mantissa}}$$

- Verification:

$$N = \left(2^{-8} + 2^{-9} + 2^{-11} + 2^{-12} + 2^{-16}\right) \times 2^{80-64} =$$
$$= \left(2^{-8} + 2^{-9} + 2^{-11} + 2^{-12} + 2^{-16}\right) \times 2^{16} =$$
$$= 2^8 + 2^7 + 2^5 + 2^4 + 2^0 =$$
$$= 256 + 128 + 32 + 16 + 1 =$$
$$= 433$$

(6.6)

§ 3.7. Power of 2 radix FLP representations

- Necessity of zones II and VI extension
- Simple mathematical connection between radix 2 and radices power of two (4,8,16) representations.

(7.1)

- General notations:

$$N = R^e \times m_R$$
$$R = 2^k$$
$$N = \left(2^k\right)^e \times m_R$$

(7.2)

- Particular notations:
  1) For $k = 2$

$$R = 2^2 = 4$$
$$N = 4^e \times m_4$$

  if normalized:

$$\frac{1}{4} \le |m_4| < 1$$

(7.3)

  2) For $k = 3$

$$R = 2^3 = 8$$

$N = 8^e \times m_8$

if normalized: (7.4)

$$\frac{1}{8} \le |m_8| < 1$$

3) For $k = 4$

$R = 2^4 = 16$

$N = 16^e \times m_{16}$

if normalized: (7.5)

$$\frac{1}{16} \le |m_{16}| < 1$$

- General rules:

1) m_R is written with 0 and 1, *but interpreted in radix R* (7.6)
2) thus m_{16} is written with 0 and 1, but interpreted in radix 16; a group of four consecutive bits associated to a hexadigit (7.7)
3) Exponent *e* is a binary integer, but $2^k <<< 16^k$! Hence, the range of expressible numbers is *significantly extended*. (7.8)

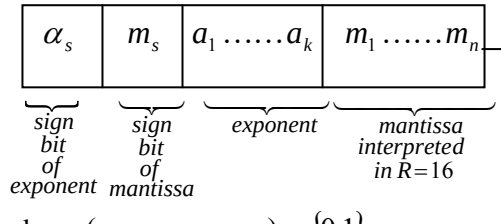
- Advantages:

1) a powerful mechanism of extension the ranges of representation
2) reduction of the probability of generation un-normalized representations
3) reduction of normalizing procedure duration (7.9)

- *Example*:

Radix $R = 16$

Format (1 , 1 , k , n) (7.10)



(7.10)

where $(\alpha_s, m_s, a_i, m_j) \in \{0,1\}$

$m_{16} \longrightarrow \underbrace{m_1 m_2 m_3 m_4}_{h_{-1}} \underbrace{m_5 m_6 m_7 m_8}_{h_{-2}} \ldots \underbrace{m_{n-3} m_{n-2} m_{n-1} m_n}_{h_{-\frac{n}{4}}}$ (7.11)

Then $m_{16} \longrightarrow h_{-1} h_{-2} \ldots h_{-\frac{n}{4}}$

Normalization condition: $h_{-1} \neq 0$ corresponds to the following cases:

0001
0010
……
……
1111

} 15 cases (7.12)

§3.8. Example of FLP representation in radix 16

- Given the decimal integer:

$N_{10} = 432$

It is required FLP representation in radix 16

$R = 16 = 2^4$

assumed FLP format: $1 + 7 + 16$

Biased exponent

$C = E + 64$ (8.1)

- Solutions:

1) Conversion of N_{10} into radix 2 and radix 16:

$432_{10} = 110110000_2 = 1B0_{16}$ (8.2)

2) Fractional representations for normalized FLP representation:

$$110110000 = 0.\ 0001\ 1011\ 0000\ 0000 \times 2^{12} = \\ = 0.\ 0001\ 1011\ 0000\ 0000 \times 16^{3} \quad (8.3)$$

3) Exponent:

$$E = 3, \quad C = 64 + 3 = 67 \longrightarrow 1000011 \quad (8.4)$$

4) FLP representation

$$\underbrace{0}_{\substack{sign\\ bit}}\ \underbrace{1000011}_{\substack{biased\\ exponent}}\ \underbrace{0001}_{\substack{h_{-1}\\ 16^{-1}}}\ \underbrace{1011}_{\substack{h_{-2}\\ 16^{-2}}}\ \underbrace{0000}_{\substack{h_{-3}\\ 16^{-3}}}\ \underbrace{0000}_{\substack{h_{-4}\\ 16^{-4}}} \quad (8.5)$$

5) Verification:

$$N_{10} = 16^{67-64}\left(1 \times 16^{-1} + B \times 16^{-2} + 0 \times 16^{-3} + \right. \\ \left. + 0 \times 16^{-4}\right) = \\ = 16^{3}\left(16^{-1} + 11 \times 16^{-2}\right) = \\ = 16^{2} + 16 \times 11 = 256 + 176 = 432 \quad (8.6)$$

6) Fractional representation for un-normalized FLP representation:

$$110110000 = 0.\ 0000\ 0000\ 0001\ 1011 \times 2^{20} = \\ = 0.\ 0000\ 0000\ 0001\ 1011 \times 16^{5} \quad (8.7)$$

7) Exponent:

$$E = 5, \quad C = 64 + 5 = 69 \longrightarrow 1000101 \quad (8.8)$$

8) FLP representation

$$\underbrace{0}_{\substack{sign\\ bit}}\ \underbrace{1000101}_{\substack{biased\\ exponent}}\ \underbrace{0000}_{\substack{h_{-1}\\ 16^{-1}}}\ \underbrace{0000}_{\substack{h_{-2}\\ 16^{-2}}}\ \underbrace{0001}_{\substack{h_{-3}\\ 16^{-3}}}\ \underbrace{1011}_{\substack{h_{-4}\\ 16^{-4}}} \quad (8.9)$$

9) Verification:

$$N = 16^{69-64}\left(0 \times 16^{-1} + 0 \times 16^{-2} + 1 \times 16^{-3} + \right. \\ \left. + B \times 16^{-4}\right) = \\ = 16^{5}\left(16^{-3} + 11 \times 16^{-4}\right) = \\ = 16^{2} + 11 \times 16^{1} = 256 + 176 = 432 \quad (8.10)$$

§3.9. Comparative analysis of representation errors

- *Maximum absolute error*:

1) Definition

$$\Delta_{\max} = \frac{distance}{2} \quad (9.1)$$

2) FXP case:

$$\Delta_{\max}^{FXP} = \frac{2^{-n}}{2} = 2^{-n-1} \quad (9.2)$$

3) FLP case:

$$\Delta_{\max}^{FLP} = \frac{2^{e-n}}{2} = 2^{e-n-1} \quad (9.3)$$

- *Relative error*:

1) Definition

$$\delta = \frac{\Delta_{\max}}{A} \quad (9.4)$$

2) FXP case:

$$\delta^{FXP} = \frac{2^{-n-1}}{A} \quad (9.5)$$

$$\delta_{\max}^{FXP} = \frac{2^{-n-1}}{A_{\min}} = \frac{2^{-n-1}}{2^{-n}} = 2^{-1} \quad (9.6)$$

$$\delta_{\min}^{FXP} = \frac{2^{-n-1}}{A_{\max}} = \frac{2^{-n-1}}{1 - 2^{-n}} \approx 2^{-n} \quad (9.7)$$

3) *Conclusion*: a *strong dependency* on the value of the number, thereby the relative error in FXP case is not constant over the range of representation. (9.8)

4) FLP case:

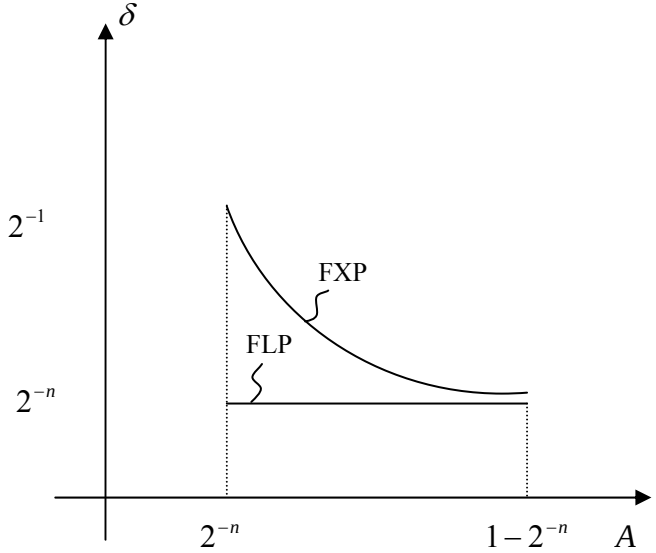
$$\delta^{FLP} = \frac{\Delta_{\max}^{FLP}}{A} = \frac{2^{e-n-1}}{2^e \cdot m} = \frac{2^{-n-1}}{m} \quad (9.9)$$

$$\delta_{\min\ n}^{FXP} = \frac{2^{-n-1}}{m_{\max}} = \frac{2^{-n-1}}{1-2^{-n}} \approx 2^{-n} \quad (9.10)$$

$$\delta_{\max\ n}^{FXP} = \frac{2^{-n-1}}{m_{\min}} = \frac{2^{-n-1}}{2^{-1}} = 2^{-n} \quad (9.11)$$

4) *Conclusion*: the relative error in normalized FLP case is *constant* over the entire range of representation. (9.12)

- Graphical representation:



(9.13)

- Comments on δ (how to improve the FXP relative errors in case of small numbers – *scaling factors*). (9.14)

§3.10. Examples of FLP representations in different computer families.

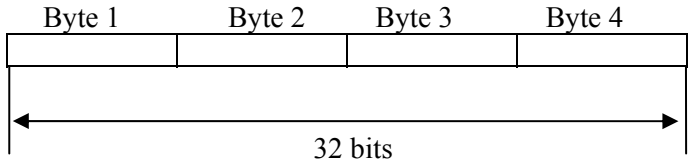
3.10.1. Felix C family

- Only normalized FLP numbers ($\frac{1}{2} \le |m| < 1$)
- Simple precision and Double precision
- Binary and hexa interpretations

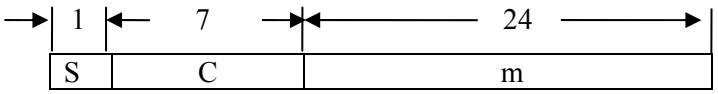
(1.1)

- *Simple precision* (***SP***) requires 4 Bytes (32 bits):

SP:



SP FLP format:



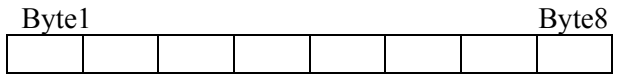
(1.2)

where: *S*= sign bit
C= biased exponent
m= mantissa satisfying the condition:

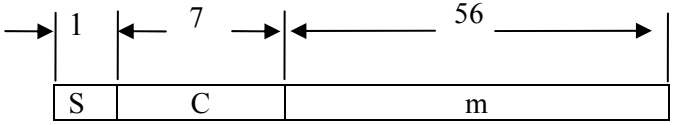
$$\frac{1}{2} \le |m| < 1$$

- *Double precision* (***DP***) requires 8 Bytes (64 bits):

DP:



DP format:



where: *S*= sign bit
C= biased exponent
m= mantissa satisfying the condition:

$$\frac{1}{2} \le |m| < 1$$

(1.3)

- For both cases the *biased exponent* C on 7 bits is represented in excess 64:

$$C = E + 64$$

(1.4)

- Ranges for *C* and *E*:

$$C \in [0, 127]$$

$$E = C - 64$$

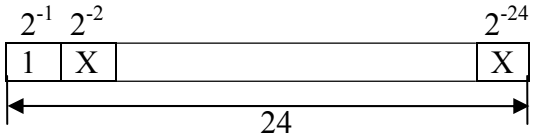
$$E_{\min} = C_{\min} - 64 = 0 - 64 = -64$$

$$E_{\max} = C_{\max} - 64 = 127 - 64 = +63$$

$$\text{Thus, } E \in [-64, 63]$$

(1.5)

- Ranges for mantissa:
  - For **SP**: mantissa represented on 24 bits, where m_{-1} is obligatory 1

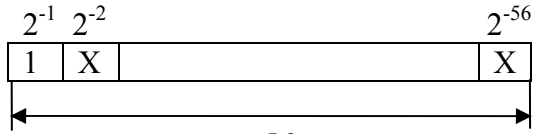


Thus,

$$|m|_{\min}^{SP} = 2^{-1}$$

$$|m|_{\max}^{SP} = 1 - 2^{-24}$$

  - For **DP**: mantissa represented on 56 bits, where m_{-1} is obligatory 1



Thus,

$$|m|_{\min}^{DP} = 2^{-1}$$

$$|m|_{\max}^{DP} = 1 - 2^{-56}$$

(1.6) (1.7)

- The ranges of representable numbers:
  - **SP**:

$$N_{\max SP}^{+} = 2^{63}(1 - 2^{-24}) \cong 2^{63}$$

$$N_{\min SP}^{+} = 2^{-64} \cdot 2^{-1} = 2^{-65}$$

$$N_{\max SP}^{-} = -2^{-64} \cdot 2^{-1} = -2^{-65}$$

$$N_{\min SP}^{-} = -2^{63}(1 - 2^{-24}) \cong -2^{63}$$

(1.8)

- **DP**:

$$N^{+}_{\max DP} = 2^{63}(1-2^{-56}) \cong 2^{63}$$
$$N^{+}_{\min DP} = 2^{-64}\cdot 2^{-1} = 2^{-65}$$
$$N^{-}_{\max DP} = -2^{-64}\cdot 2^{-1} = -2^{-65}$$
$$N^{-}_{\min DP} = -2^{63}(1-2^{-56}) \cong -2^{63}$$

(1.9)

- *Example*:

$N_{10} = 433$

$433_{10} = 110110001_2$

Fractional notation of N

$110110001 = 0.110110001 \times 2^9$

Biased exponent

$E = 9 \rightarrow C = 9 + 64 = 73 \rightarrow 1001001$

In SP representation mantissa consists of 24 bits

Required FLP representation:

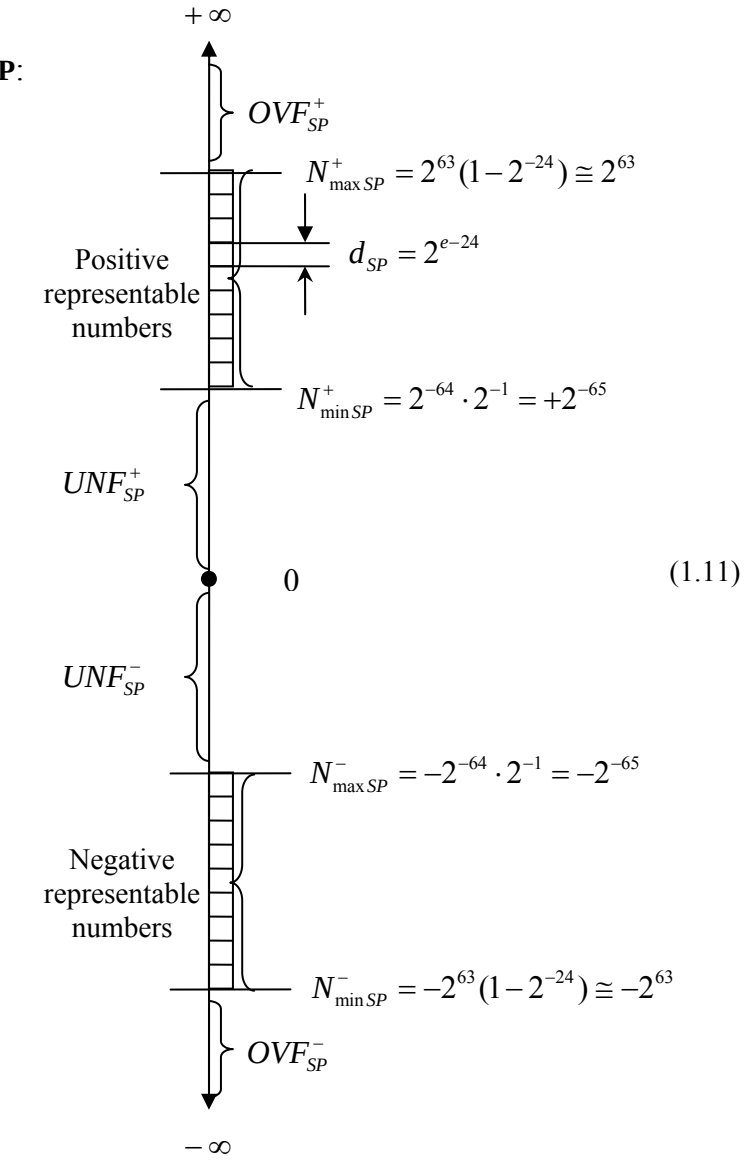
0 1001001 11011000100000000000000

sign — Biased exponent — mantissa

(1.10)

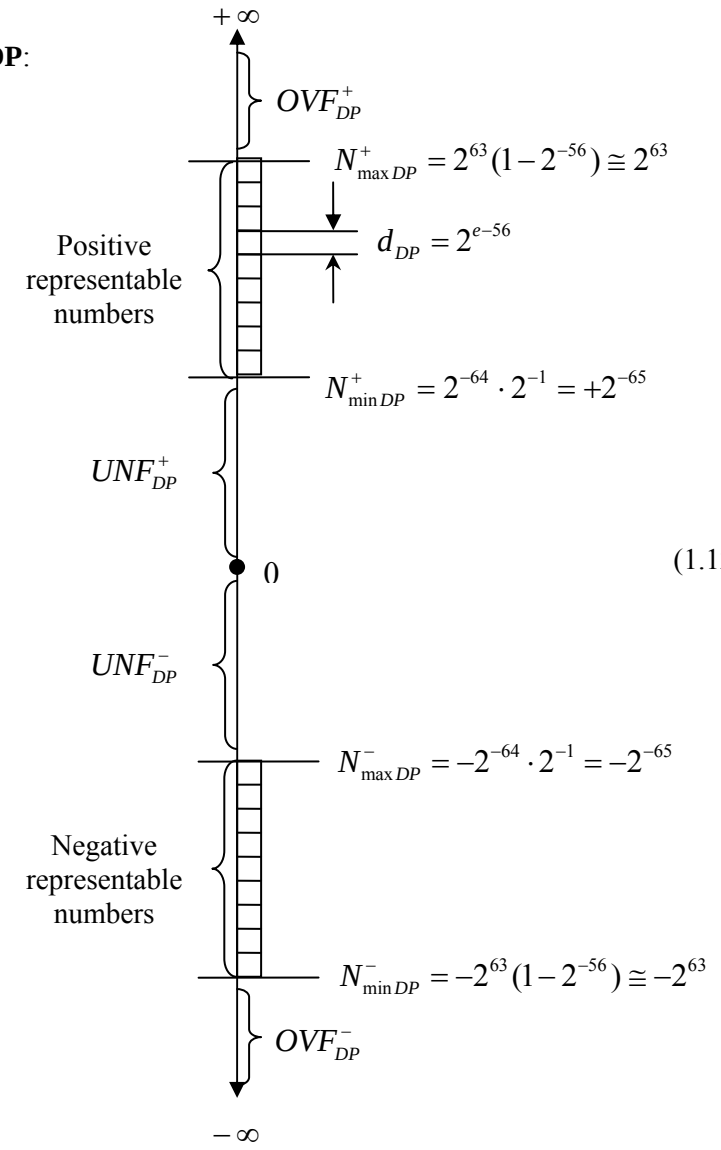
- Representation on the real axis:

SP:



(1.11)

DP:



(1.12)

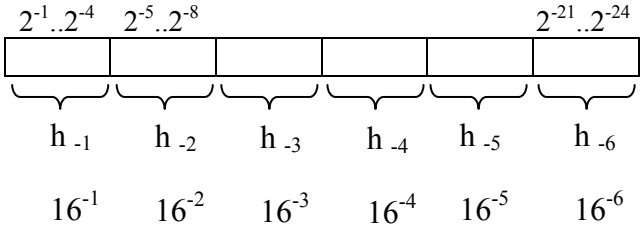
- *Conclusion*: the ranges are quite identical, but the number of representable numbers is much greater in DP case ($d_{DP} < d_{SP}$) } (1.13)
- General representation in *base 16*:

  $N = 16^{e} \cdot m = (2^{4})^{e} \cdot m$

  For normalised case:

  $\frac{1}{16} \le | m | < 1$ } (1.14)

- Mantissa is interpreted in base 16, as follows:
- For **SP**:



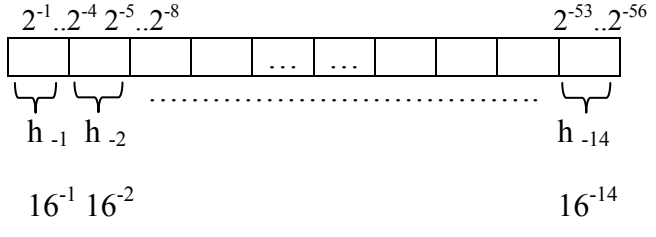
(1.15)

  - The extreme values are:

    $| m |_{\min SP} = 16^{-1}$

    $| m |_{\max SP} = 1 - 16^{-6}$ } (1.16)

- For **DP**:



(1.17)

  - The extreme values are:

    $| m |_{\min DP} = 16^{-1}$

    $| m |_{\max DP} = 1 - 16^{-14}$ } (1.18)

- *Biased exponent* remains the same:
  $E \in [-64, +63]$ (1.19)

- Ranges of representation:

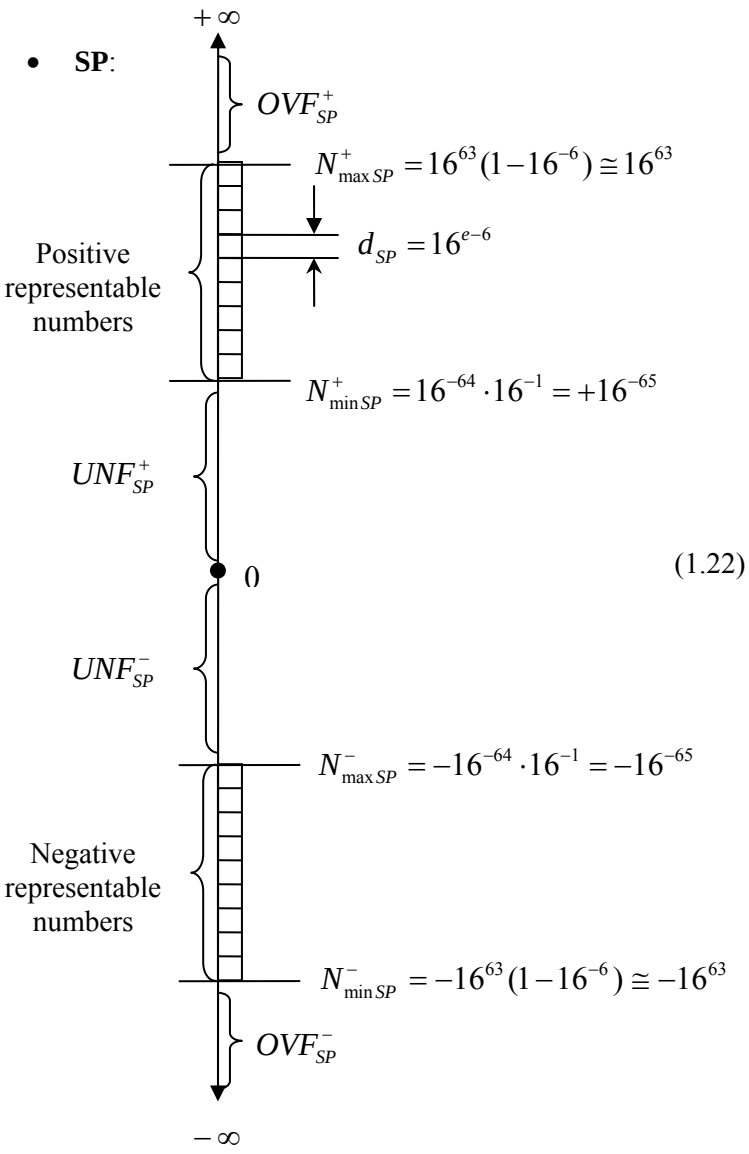
  - **SP**:

$$N^{+}_{\max SP} = 16^{+63}(1-16^{-6}) \cong 16^{+63}$$
$$N^{+}_{\min SP} = 16^{-64} \cdot 16^{-1} = 16^{-65}$$
$$N^{-}_{\max SP} = -16^{-64} \cdot 16^{-1} = -16^{-65}$$
$$N^{-}_{\min SP} = -16^{63}(1-16^{-6}) \cong -16^{63} \quad (1.20)$$

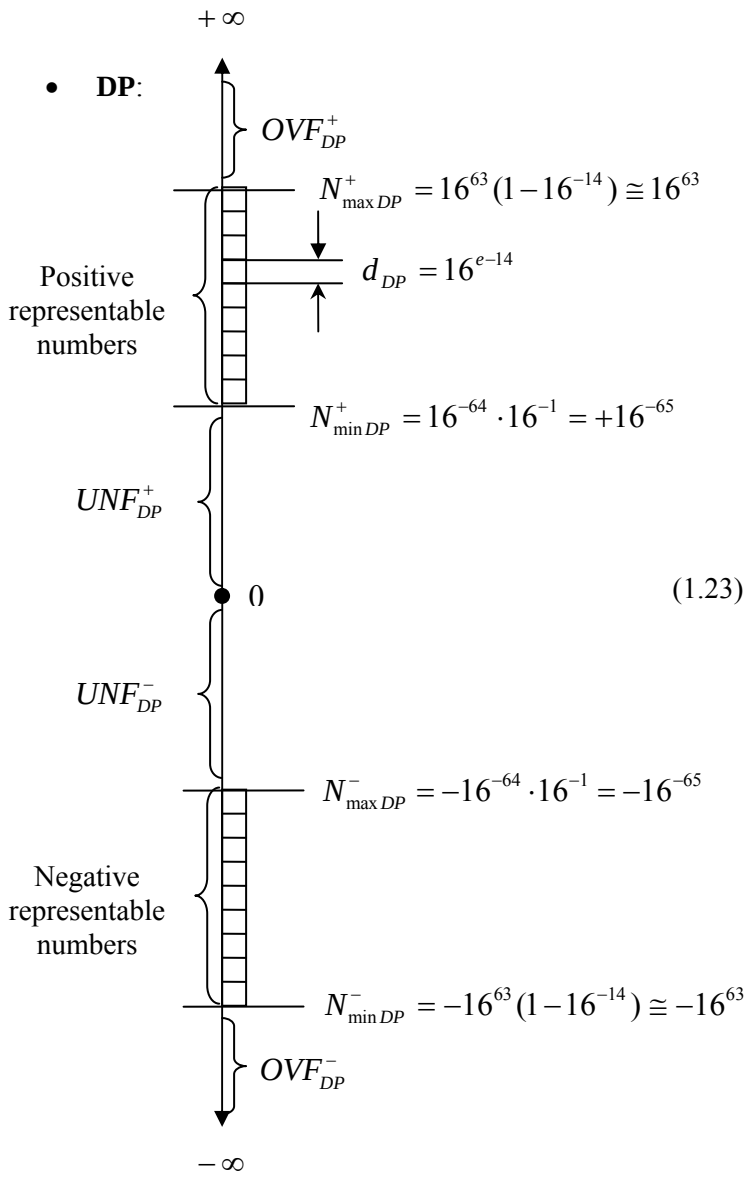
  - **DP**:

$$N^{+}_{\max DP} = 16^{+63}(1-16^{-14}) \cong 16^{+63}$$
$$N^{+}_{\min DP} = 16^{-64} \cdot 16^{-1} = 16^{-65}$$
$$N^{-}_{\max DP} = -16^{-64} \cdot 16^{-1} = -16^{-65}$$
$$N^{-}_{\min DP} = -16^{63}(1-16^{-14}) \cong -16^{63} \quad (1.21)$$

- Representation on the real axis:



(1.22)



(1.23)

- *Conclusions*:

1) the number of expressible numbers is identical with that corresponding to interpretation in base 2, but the ranges were expanded, having in view that : (1.24)

$$16^{e} >> 2^{e}$$

2) the accuracy is diminished since the distance between nodes is greater : (1.25)

SP: $d_{SP\ 2} = 2^{e-24}$

$d_{SP\ 16} = 16^{e-6} = 2^{4e-24} >> d_{SP\ 2}$ (1.26)

DP: $d_{DP\ 2} = 2^{e-56}$

$d_{DP\ 16} = 16^{e-14} = 2^{4e-56} >> d_{DP\ 2}$ (1.27)

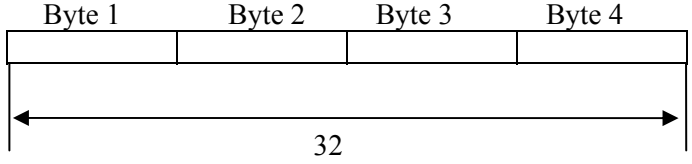
3.10.2. PDP-11 Family

- Only normalized FLP numbers ($\frac{1}{2} \le |m| < 1$)
- Simple precision and Double precision representations

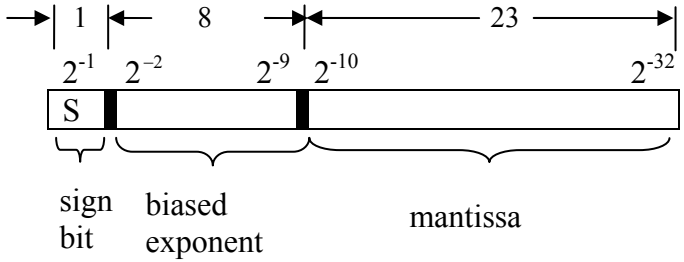
(2.1)

- Only binary interpretation
- *Simple precision* (**SP**) representation on 4 Bytes (32 bits):

SP



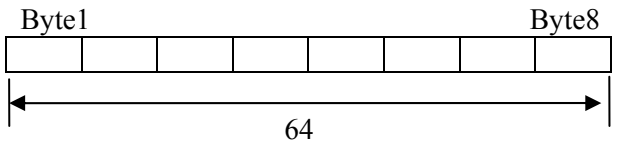
SP format:



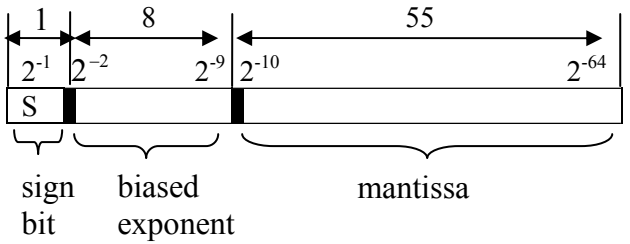
(2.2.)

- *Double precision* (**DP**) representation on 8 Bytes (64 bits):

DP:



DP format:



(2.3.)

- For both cases the *biased exponent C* on 8 bits is represented in excess 128 :

$$C = E + 128$$

(2.4)

- Ranges for C and E :

$$C \in [0,255]$$

$$E = C - 128$$

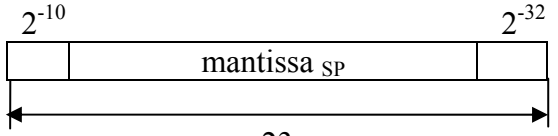
$$E_{min} = C_{min} - 128 = 0 - 128 = -128$$

$$E_{max} = C_{max} - 128 = 255 - 128 = +127$$

Thus, $E \in [-128,127]$

(2.5)

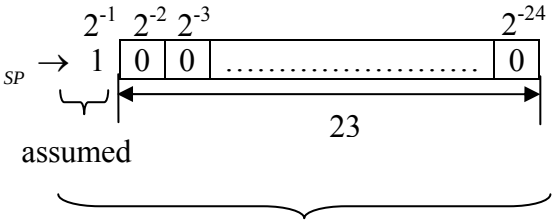
- Ranges for mantissa :
  - For **SP** case mantissa is represented on 23 bits:



(2.6)

But, because only normalized mantissas are allowed, the first bit, m_{-1}, is always 1, so that it could be skipped, being *assumed*. Then, with 23 bits there are represented mantissas on 24 bits. (2.7)

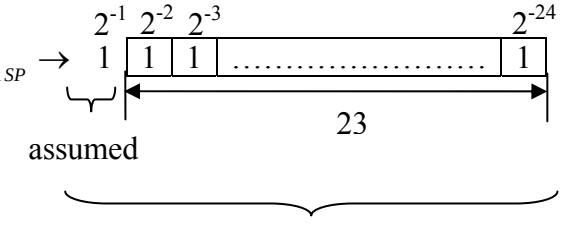
- $|m|_{\min\ SP} \rightarrow$



(2.8)

$$|m|^{SP}_{\min} = 2^{-1}$$

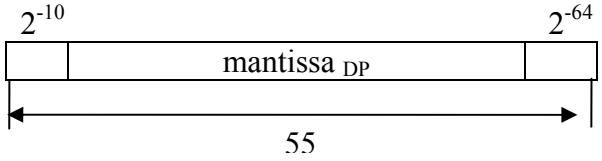
- $|m|_{\max\ SP} \rightarrow$



(2.9)

$$|m|^{SP}_{\max} = 1 - 2^{-24}$$

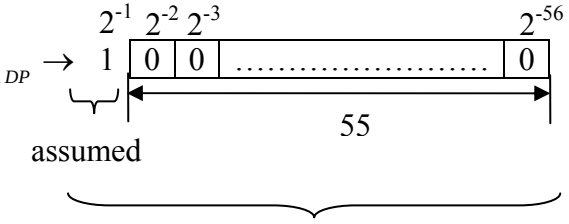
- For **DP** case mantissa is represented on 55 bits:



(2.10)

- Similarly, the most significant bit is always 1,so that it is not represented anymore, being assumed. Therefore, with 55 bits there are represented mantissas on 56 bits. (2.11)

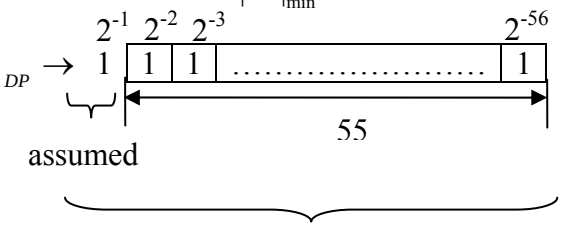
- $|m|_{\min\ DP} \rightarrow$



(2.12)

$$|m|^{DP}_{\min} = 2^{-1}$$

- $|m|_{\max\ DP} \rightarrow$



(2.13)

$$|m|^{DP}_{\max} = 1 - 2^{-56}$$

- The ranges of representable numbers :
  - **SP** : $N^{+}_{\max\ SP} = 2^{127}(1-2^{-24}) \cong 2^{127} \cong 10^{38}$

    $N^{+}_{\min\ SP} = 2^{-128} \cdot 2^{-1} = 2^{-129}$

    $N^{-}_{\max\ SP} = -2^{-128} \cdot 2^{-1} = -2^{-129}$

    $N^{-}_{\min\ SP} = -2^{127}(1-2^{-24}) \cong -2^{127} \cong -10^{38}$

    (2.14)
  - **DP**: $N^{+}_{\max\ DP} = 2^{127}(1-2^{-56}) \cong 2^{127}$

    $N^{+}_{\min\ DP} = 2^{-128} \cdot 2^{-1} = 2^{-129}$

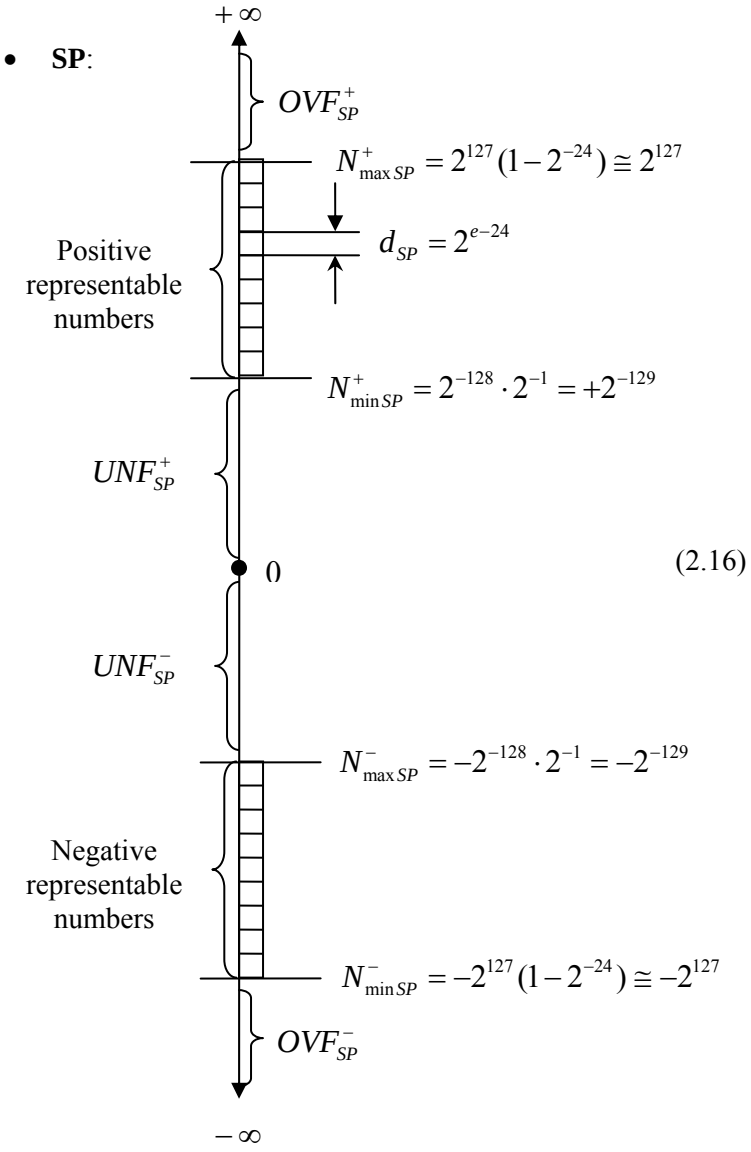
    $N^{-}_{\max\ DP} = -2^{-128} \cdot 2^{-1} = -2^{-129}$

    $N^{-}_{\min\ DP} = -2^{127}(1-2^{-56}) \cong -2^{127}$

    (2.15)

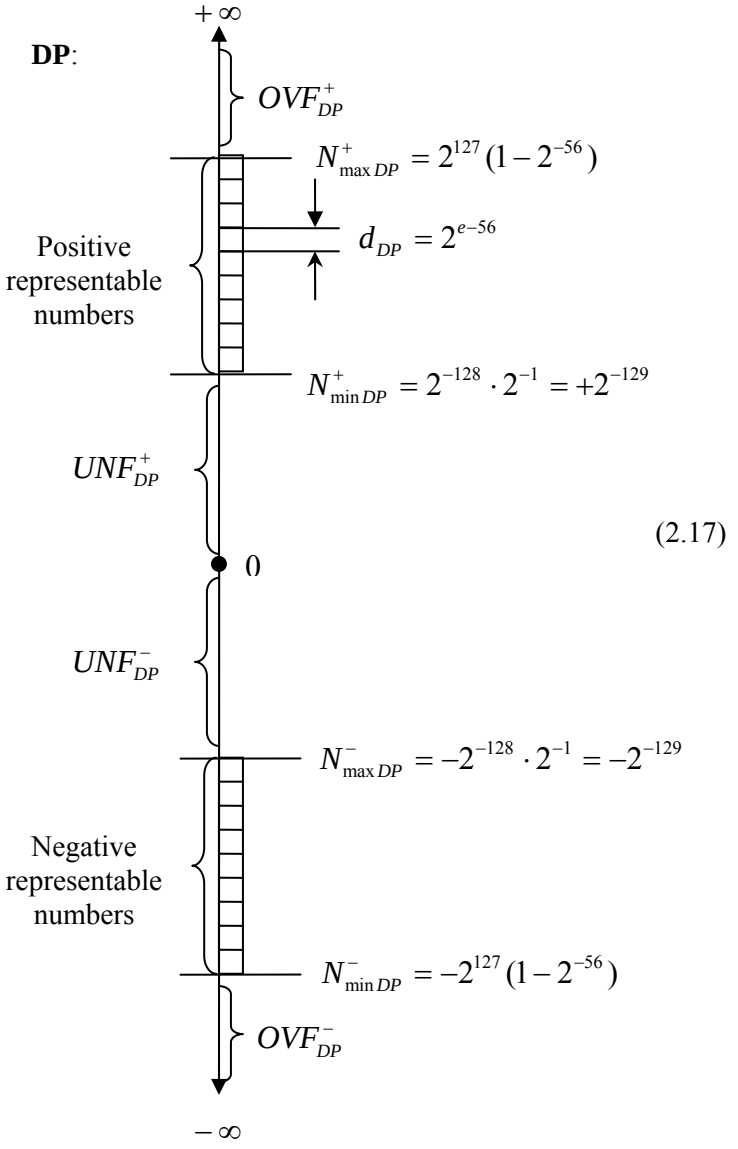
- Representation on the real axis :

- **SP**:



(2.16)

- **DP**:



(2.17)

§3.11. IEEE Floating Point Standard 754 - 85

3.11.1 Short History

- Until about 1980 each computer manufacturer had its own FLP format .Some of them did arithmetic operations incorrectly and there was no data portability.
- IEEE set up a committee to standardize FLP arithmetic.
- Two main advantages by standardization:
  a) data can be exchanged between different computers
  b) hardware designers have a unique set of specifications when designing CPUs
- In 1985 it was adopted IEEE standard 754 for FLP representations.
- All big companies producing microprocessors complied to this standard (Intel , Motorola , MIPS , SPARC etc)
- The standard encompasses the advanced experience of many well-known CPU/micro manufacturers.
- The standard was developed to facilitate the portability of programs from one processor to another processor.

(11.1.1)

3.11.2 Basic FLP Formats

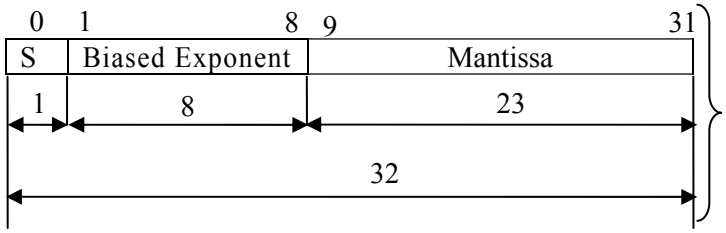
- Three FLP formats :
  1) *Single Precision* (SP) on 32 bits.
  2) *Double Precision* (DP) on 64 bits.
  3) *Extended Precision* (EP) on 80 bits.

(11.2.1)

- *Extended precision* is used primarily inside FLP ALUs. Its role is to reduce errors in case of the *rounding* operations. Also, EP lessens the chance of an intermediate overflow to abort computations. This format is not available to the users. EP includes additional bits both at exponent and mantissa fields. EP used strictly for *intermediate calculations,* because this format lessens the chance of a final result to be contaminated by excessive round off error.
  Since EP is of interest only for designers of CPUs, in what follows, it will not be discussed.
  The focus of presentation will be the *Single Precision* (SP) and *Double Precision* (DP) formats.

(11.2.2)

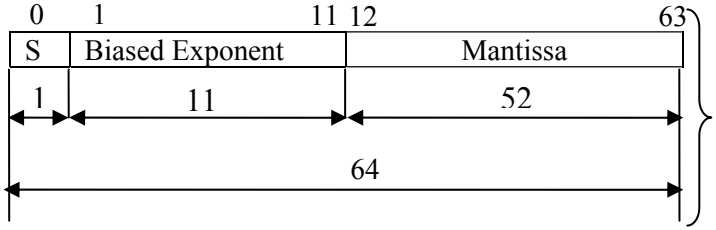
- *Single precision* format :



(11.2.3)

SP format is on 32 bits with an 8 bit segment for the biased exponent.

- *Double precision* format :



(11.2.4)

DP format is on 64 bits with an 11 bit segment for the biased exponent.

- S = sign bit of the represented number
  S=0 for N>0
  S=1 for N<0

(11.2.5)

- *Biased exponents* are different for SP and DP :
  1) In SP exponent uses excess 127 representation

$$C_{SP} = E_{SP} + 127$$

(11.2.6)

  2) In DP exponent uses excess 1023 representation

$$C_{DP} = E_{DP} + 1023$$

(11.2.7)

- Range of exponents :

1) In **SP** :

$$C_{SP} = E_{SP} + 127 \rightarrow E_{SP} = C_{SP} - 127$$
$$C_{SP} \in [0,255], \text{ the range for 8 bit numbers} \quad (11.2.8)$$

The extreme values for C_{SP} :

$$C_{SP\min} \rightarrow 00000000 \rightarrow 0$$
$$C_{SP\max} \rightarrow 11111111 \rightarrow 255 \quad (11.2.9)$$

The extreme values for E_{SP}

$$E_{SP\min} = C_{SP\min} - 127 = 0 - 127 = -127$$
$$E_{SP\max} = C_{SP\max} - 127 = 255 - 127 = +128 \quad (11.2.10)$$

Thus, $E_{SP} \in [-127,+128]$

2) In **DP** :

$$C_{DP} = E_{DP} + 1023 \rightarrow E_{DP} = C_{DP} - 1023$$
$$C_{DP} \in [0,2047], \text{ the range of 11 bit numbers} \quad (11.2.11)$$

The extreme values for C_{DP} :

$$C_{DP\min} \rightarrow 00000000000 \rightarrow 0$$
$$C_{DP\max} \rightarrow 11111111111 \rightarrow 2047 \quad (11.2.12)$$

The extreme values for E_{DP} :

$$E_{DP\min} = C_{DP\min} - 1023 = 0 - 1023 = -1023$$
$$E_{DP\max} = C_{DP\max} - 1023 = 2047 - 1023 =$$
$$= +1024 \quad (11.2.13)$$

Thus, $E_{DP} \in [-1023,+1024]$

- Mantissa is always interpreted in base 2 (never a power of 2 base) (11.2.14)
- One of the traditional problems with FLP numbers is how to deal with **UNF**,**OVF** or uninitialized numbers. IEEE 754 standard deals with these problems explicitly by adding to the traditional normalized numbers other four numerical types (11.2.15)

- Therefore the following numerical types were designed: (11.2.16)
  a) *Normalized numbers*
  b) *Denormalized numbers*
  c) *Zero*
  d) *Infinity*
  e) *Not a Number (NaN)*

3.11.3 Normalized Numbers

- The extreme values of the biased exponent are not allowed for normalized numbers, being assigned to other FLP representations.
- Therefore, the allowed biased exponent values, designated C^N, are in the ranges: (11.3.1)
  - $C_{SP}^N \in (0,255)$: $\boxed{00000001}$ $C_{SP_{min}}^N$ ⋮ $\boxed{11111110}$ $C_{SP_{max}}^N$
  - $C_{DP}^N \in (0,2047)$: $\boxed{00000000001}$ $C_{DP_{min}}^N$ ⋮ $\boxed{11111111110}$ $C_{DP_{max}}^N$
- According to definition, a normalized number fraction begins with a binary point, followed by a 1, and the rest of the fraction.
- As in case of PDP-11 computers, the leading 1 in the fraction does not have to be stored, its presence is *assumed*
- Standard IEEE 754 uses the principle of an *implicit 1*. (11.3.2)

- *Significand* is a concept used instead of the words fraction or mantissa; it is designated *S* and it is composed of:
  a)the implied leading 1, but placed in the integer position 2^0.
  b)the binary point
  c)23 (**SP**) or 52 (**DP**) arbitrary bits representing the mantissa (fraction) (11.3.3)

$$S \rightarrow 1.\underbrace{xx.....x}_{23} \rightarrow \text{in SP} \quad (11.3.4)$$

$$S \rightarrow 1.\underbrace{xx.....x}_{52} \rightarrow \text{in DP} \quad (11.3.5)$$

- *Conclusion*: all normalized numbers have a significand *S* in the range (11.3.6)

$$1 \le S < 2$$

- Detailed calculation of the range of significands

1) for **SP**:

$$|S_{\min SP}| = 1.\underbrace{00.....0}_{23} \rightarrow 1 \quad (11.3.7)$$

$$|S_{\max SP}| = 1.\underbrace{11.....1}_{23} \rightarrow 1 + \left(1 - 2^{-23}\right) = 2 - 2^{-23} \quad (11.3.8)$$

Hence, $1 \le |S_{SP}| \le 2 - 2^{-23}$ (11.3.9)

$$1 \le |S_{SP}| < 2 \quad (11.3.10)$$

2) for **DP**:

$$|S_{\min DP}| = 1.\underbrace{00.....0}_{52} \rightarrow 1 \quad (11.3.11)$$

$$|S_{\max DP}| = 1.\underbrace{11.....1}_{52} \rightarrow 1 + (1 - 2^{-52}) = 2 - 2^{-52} \quad (11.3.12)$$

Hence, $1 \le |S_{DP}| \le 2 - 2^{-52}$ (11.3.13)

$$1 \le |S_{DP}| < 2 \quad (11.3.14)$$

- Ranges of normalized number representations:

1) for **SP**:

$C_{SP}^N \in (0,255)$, or $0 < C_{SP}^N < 255$, or $1 \le C_{SP}^N \le 254$ (11.3.15)

$E_{SP}^N \in (-127,+128)$, or $-127 < E_{SP} < 128$, or $-126 \le E_{SP}^N \le +127$ (11.3.16)

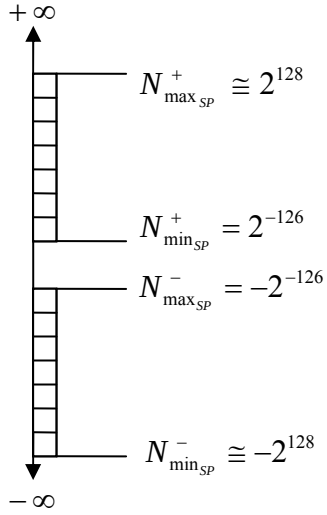
$|S_{\min SP}| = 1$

$|S_{\max SP}| = 2\text{-}2^{-23}$ (11.3.17)

Then,

$$N_{\max SP}^{+} = 2^{+127} \times (2\text{-}2^{-23}) \cong 2^{128}$$
$$N_{\min SP}^{+} = 2^{-126} \times 1 = 2^{-126}$$
$$N_{\max SP}^{-} = -2^{-126} \times 1 = -2^{-126}$$
$$N_{\min SP}^{-} = -2^{+127}(2\text{-}2^{-23}) \cong -2^{128} \quad (11.3.18)$$

- Representation on the real axis:



(11.3.19)

2) for **DP**:

$$C_{DP}^{N} \in (0,2047),\ \text{or}\ 0 < C_{DP}^{N} < 2047,\ \text{or}\ 1 \le C_{DP}^{N} \le 2046 \quad (11.3.20)$$

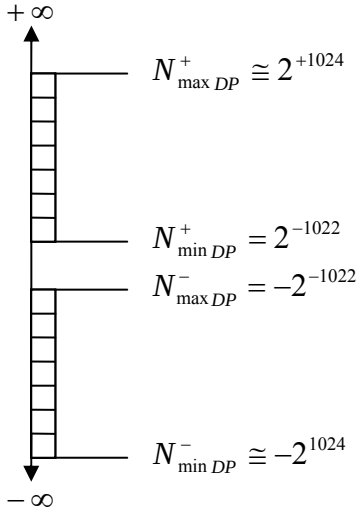
$$E_{DP}^{N} \in (-1023,+1024),\ \text{or}\ -1023 < E_{DP} < 1024,\ \text{or}\ -1022 \le E_{DP}^{N} \le 1023 \quad (11.3.21)$$

$$\left|S_{\min DP}\right| = 1 \qquad \left|S_{\max DP}\right| = 2 - 2^{-52} \quad (11.3.22)$$

Then,

$$N_{\max DP}^{+} = 2^{1023}\times(2\text{-}2^{-52}) \cong 2^{+1024}$$
$$N_{\min DP}^{+} = 2^{-1022}\times 1 = +2^{-1022}$$
$$N_{\max DP}^{-} = -2^{-1022}\times 1 = -2^{-1022}$$
$$N_{\min DP}^{-} = -2^{1023}\times(2\text{-}2^{-52}) \cong -2^{-1024} \quad (11.3.23)$$

- Representation on the real axis:



(11.3.24)

3.11.4. Examples of normalized number representations.

- *Example Nr.* 1 :

It is given a decimal fraction N_{10}=+0.5 and it is required its FLP representation.

$$N_2 = +0.1$$
$$N_2 = (1.0)\times 2^{-1} \rightarrow E = -1$$
$$C = E + 127 = -1 + 127 = 126$$
$$C = 01111110$$
$$S = 1.\underbrace{000.....0}_{23}$$
$$N > 0 \rightarrow \text{Sign} = 0$$

SP FLP representation: 23 bits

0	1							8	9															31
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3			F				0	0			0				0				0				0

Thus, $N_{10} = +0.5 \rightarrow$ 3F000000h (11.4.1)

- *Example Nr.2:*

It is given the decimal integer N_{10}=+1 and it is required its FLP representation.

$N_2 = 1$

$N_2 = (1.0)\times 2^0 \rightarrow E = 0$

$C = \text{E+127=0+127=127}$

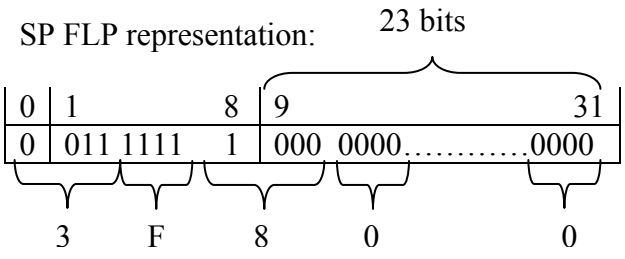
$C = 01111111$

$S = 1.\underbrace{00.....0}_{23}$

$N > 0 \rightarrow \text{Sign} = 0$

(11.4.2)

SP FLP representation:



Thus, $N = +1 \rightarrow$ 3F800000h

- *Example Nr.3:*

It is given the decimal number N_{10}=1.5 and it is required its FLP representation.

$N_2 = 1.1$

$N_2 = (1.1)\times 2^0 \rightarrow E = 0$

$C = \text{E+127=0+127=127}$

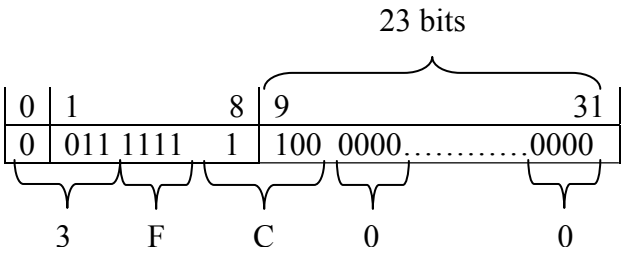
$C = 01111111$

$S = 1.\underbrace{1000.....0}_{23}$

$N > 0 \rightarrow \text{Sign} = 0$

SP FLP representation:

(11.4.3)



Thus, $N_{10} = +1.5 \rightarrow$ 3FC00000h

3.11.5. Denormalized numbers

- In general case of FLP representations there were defined the extreme regions **OVF**$^{+/-}$, **UNF**$^{+/-}$. (11.5.1)

- *Denormalized numbers* were introduced to solve the case of $\mathbf{UNF}^{+/-}$, that is when the result of a calculation has a magnitude smaller then the smallest normalized FLP number that can be represented:

$$\mathbf{UNF}^{+}: \mathrm{N} < \mathrm{N}^{+}_{\min SP},\ \mathrm{N} < \mathrm{N}^{+}_{\min DP}$$

$$\mathbf{UNF}^{-}: \mathrm{N} > \mathrm{N}^{-}_{\max SP},\ \mathrm{N} > \mathrm{N}^{-}_{\max DP} \qquad (11.5.2)$$

- Previously, such cases were solved either by taking the result 0 and continuing calculations, or causing an FLP *underflow trap*. (11.5.3)
- Since neither of these is satisfactory, IEEE Standard 754 invented *denormalized numbers*(N^D) (11.5.4)

- Rules of representation:

a) N^D has the *biased exponent 0*,

$$C^D = 0 \rightarrow E^D_{SP} = -127,\ \mathrm{E}^D_{DP} = -1023 \qquad (11.5.5)$$

b) *Significand* is replaced by the traditional mantissa (fraction) of 23 or 52 bits, so that the implied integer 1 vanishes.

$$|m|_{\max SP} = 0.\underbrace{11.....1}_{23} = 1 - 2^{-23}$$

$$|m|_{\max DP} = 0.\underbrace{11.....1}_{52} = 1 - 2^{-52} \qquad (11.5.6)$$

- The ranges of mantissas:

1) for **SP** *:* $2^{-23} \le |m|_{SP} \le 1\text{-}2^{-23}$

$$2^{-23} \le |m|_{SP} < 1$$

2) for **DP** *:* $2^{-52} \le |m|_{DP} \le 1\text{-}2^{-52}$

$$2^{-52} \le |m|_{DP} < 1 \qquad (11.5.7)$$

- Ranges of denormalized numbers:

1) for **SP** *:*

$$N^D_{max_{SP+}} = +2^{-127} \times (1 - 2^{-23}) \cong 2^{-127}$$

$$N^D_{min_{SP+}} = +2^{-127} \times 2^{-23} = 2^{-150}$$

$$N^D_{max_{SP-}} = -2^{-127} \times 2^{-23} = -2^{-150}$$

$$N^D_{min_{SP-}} = -2^{-127} \times (1 - 2^{-23}) \cong -2^{-127} \qquad (11.5.8)$$

2) for **DP** *:*

$$N^D_{max_{DP+}} = +2^{-1023} \times (1 - 2^{-52}) \cong 2^{-1023}$$

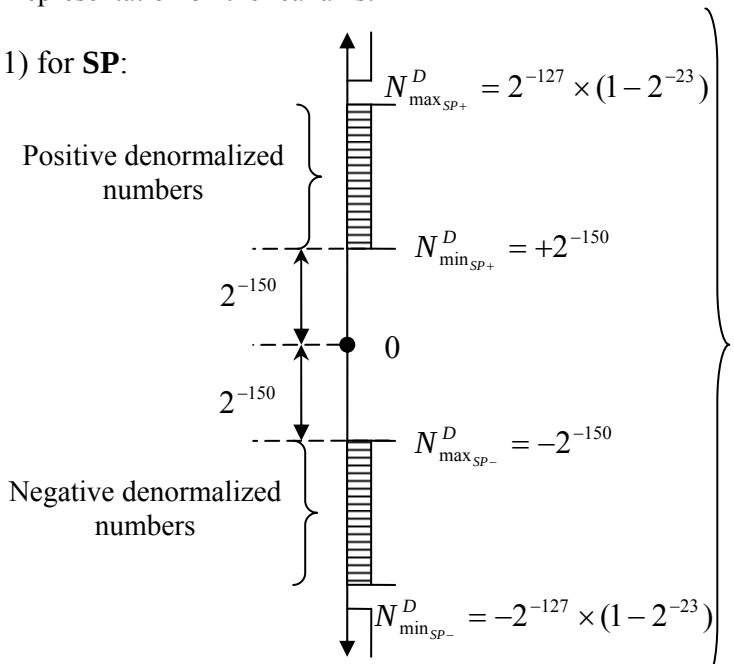
$$N^D_{min_{DP+}} = +2^{-1023} \times 2^{-52} = 2^{-1075}$$

$$N^D_{max_{DP-}} = -2^{-1023} \times 2^{-52} = -2^{-1075}$$

$$N^D_{min_{DP-}} = -2^{-1023} \times (1 - 2^{-52}) \cong -2^{-1023} \qquad (11.5.9)$$

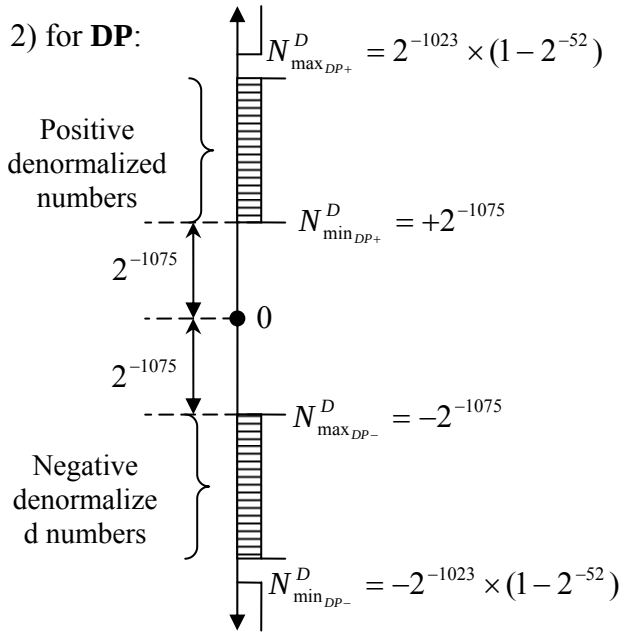
- Representation on the real axis:

1) for **SP**:



(11.5.10)

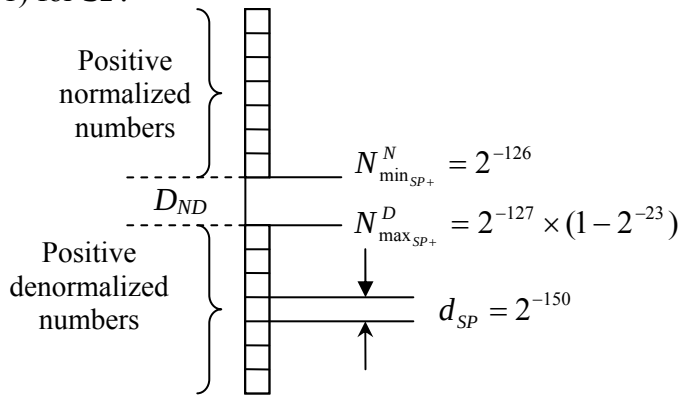
2) for **DP**:



(11.5.11)

- Contact region between *Normalized* and *Denormalized* numbers:

1) for **SP**:



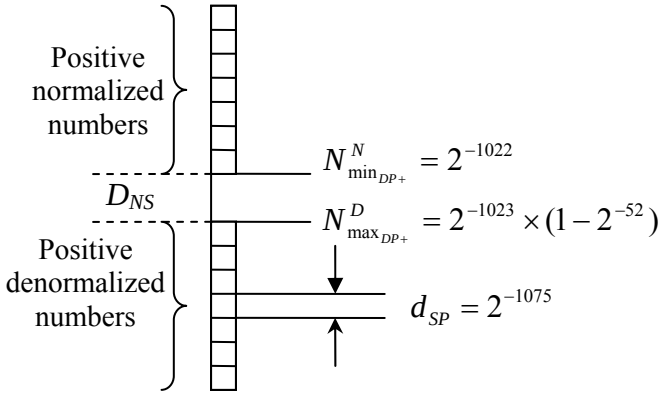
(11.5.12)

$$
\begin{aligned}
Distance_{ND} &= 2^{-126} - 2^{-127} \times (1 - 2^{-23}) = \\
&= 2^{-126} - 2^{-127} + 2^{-150} = \\
&= 2^{-127}(2^1 - 1) + 2^{-150} = \\
&= 2^{-127} + 2^{-150}
\end{aligned}
\qquad (11.5.13)
$$

- *Distance* in denormalized region:

$$
\begin{aligned}
&N^D_1 = 2^{-127} \times m_{SP} \\
&N^D_2 = 2^{-127} \times (m_{SP} + 2^{-23}) \\
&d_{SP} = N^D_2 - N^D_1 = 2^{-127}(m_{SP} + 2^{-23}) - 2^{-127} \cdot m_{SP} = \\
&\quad = 2^{-127} \cdot m_{SP} + 2^{-150} - 2^{-127} \cdot m_{SP} = \\
&\quad = 2^{-150}
\end{aligned}
\qquad (11.5.14)
$$

2) for **DP**:



(11.5.15)

$$
\begin{aligned}
Distance_{ND} &= 2^{-1022} - 2^{-1023} \cdot (1 - 2^{-52}) = \\
&= 2^{-1022} - 2^{-1022} + 2^{-1075} = \\
&= 2^{-1023} \cdot (2^1 - 1) + 2^{-1075} = \\
&= 2^{-1023} + 2^{-1075}
\end{aligned}
\qquad (11.5.16)
$$

- *Distance* in denormalized region:

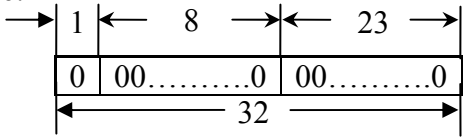
$$\left.\begin{aligned} N_1^D &= 2^{-1023}\cdot m_{DP} \\ N_2^D &= 2^{-1023}\cdot (m_{DP}+2^{-52}) \\ \\ d_{DP} &= N_2^D - N_1^D = \\ &= 2^{-1023}\cdot (m_{DP}+2^{-52}) - 2^{-1023}\cdot m_{DP} = \\ &= 2^{-1023}\cdot m_{DP} + 2^{-1075} - 2^{-1023}\cdot m_{DP} = \\ &= 2^{-1075} \end{aligned}\right\} \quad (11.5.17)$$

3.11.6. Representation of Number 0

- There are defined two zeroes: 0^+ and 0^-.
- The configurations:

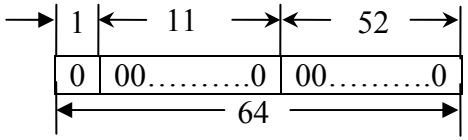
a) Positive zero:

1) for **SP**



(11.6.1)

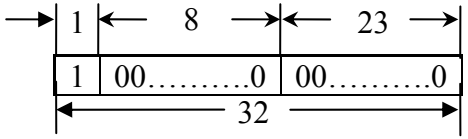
2) for **DP**



(11.6.2)

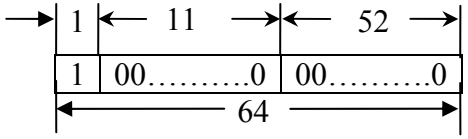
b) Negative zero:

1) for **SP**



(11.6.3)

2) for **DP**



(11.6.4)

- Rules of representation of ± 0:

$$\left\{\begin{aligned} &C_{\min} = 0 \\ &|m| = 0 \\ &Sign = 0 \rightarrow 0^+ \\ &Sign = 1 \rightarrow 0^- \end{aligned}\right. \quad (11.6.5)$$

3.11.7. Representation of infinity

- It was defined a positive overflow **OVF**$^+$ when
$$N > N^N_{\max +}$$
- It was defined a negative overflow **OVF**$^-$ when
$$N < N^N_{\min -}$$

(11.7.1)

- In general, emergence of **OVF**$^+$ or **OVF**$^-$ constitutes a singularity, which is signalled by generating an error signal.

(11.7.2)

- In IEEE FLP standard 754-85 it was adopted a more flexible solution:
  - **OVF**$^+$ is defined as $+\infty$
  - **OVF**$^-$ is defined as $-\infty$

(11.7.3)

- On $\pm\infty$ there are defined *valid operations*:

$$5+(+\infty)=+\infty \qquad 5-(+\infty)=-\infty$$
$$5-(-\infty)=+\infty \qquad \infty-(-\infty)=+\infty$$
$$5\times(\infty)=\infty \qquad -\infty+(-\infty)=-\infty$$
$$(+\infty)+(+\infty)=+\infty \qquad 5:(\pm\infty)=0$$

(11.7.4)

- Rules for representation of $\pm\infty$:

  Biased Exponent $\rightarrow C_{max}$

  Mantissa field $\rightarrow 0$ (which is not allowed for normalized numbers)

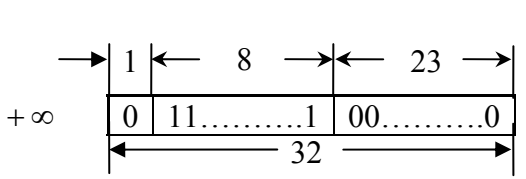
  Sign = 0 $\rightarrow$ $+\infty$

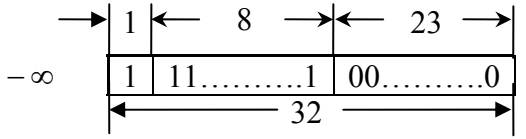
  Sign = 1 $\rightarrow$ $-\infty$

(11.7.5)

- The configurations:

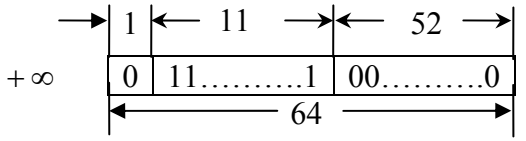
1) **SP**:

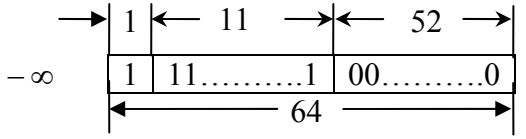




(11.7.6)

2) **DP**:





(11.7.7)

3.11.8. Not a Number (NaN) representation.

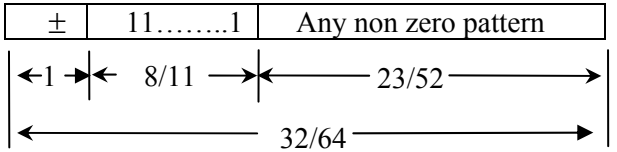
- NaN represents situations when results of arithmetic operations are *not defined*. (11.8.1)
- NaN is a symbolic entity encoded in FLP format of which there are two types: (11.8.2)
  a) signalling
  b) quiet
- The *signalling* NaN shows an invalid operation exception, whenever it appears as operand. (11.8.3)
- The *quiet* NaN propagates through almost any arithmetic operations without signaling an exception. (11.8.4)
- Table with *quite* NaN:

Operation	Example
Addition/Subtraction	$(+\infty)+(-\infty)$ $(-\infty)+(+\infty)$ $(+\infty)-(+\infty)$ $(-\infty)-(-\infty)$
Multiplication	$0\times(\pm\infty)$
Division	$\frac{0}{0}, \frac{\infty}{\infty}$
Square Root	$\sqrt{x}$, where $x<0$

(11.8.5)

- Thus, NaN can be used in FLP format as an operand with predictable results. (11.8.6)
- Rules for representation of NaN in IEEE 754 standard: (11.8.7)
  Biased Exponent: C_{max}
  Mantissa: any non zero bit pattern

- Configuration



(11.8.8)

- *Observation*: the mantissa value can be used to distinguish between a quiet NaN and a signaling NaN; also it can specify particular exception conditions. (11.8.9)

3.11.9. Recapitulation of all IEEE FLP standard 754-85 numerical types

1) *Normalized numbers*:

±	0<EXP<Max	Any pattern

2) *Denormalized numbers*:

±	0	Any nonzero pattern

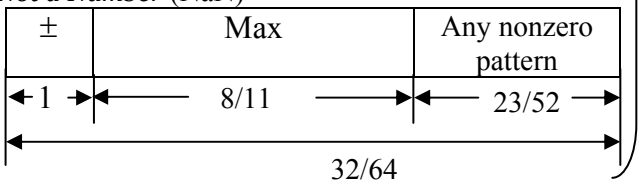
3) *Zero*:

±	0	0

4) *Infinity*:

±	Max	0

5) *Not a Number* (NaN)

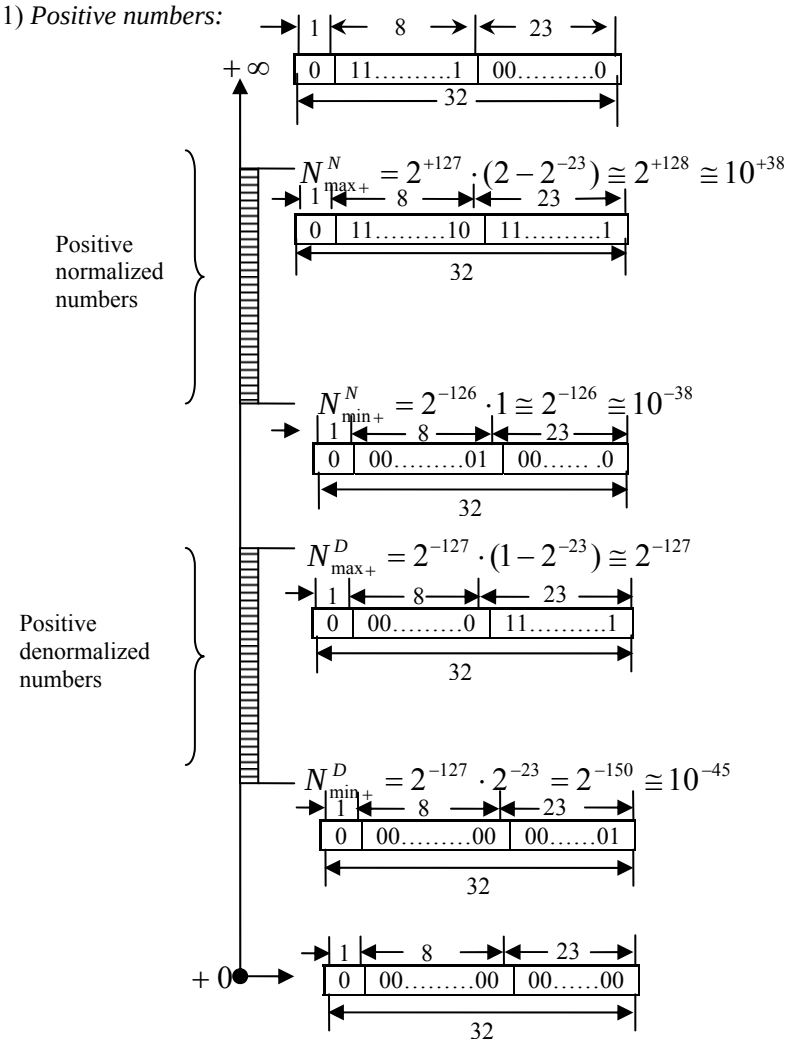


(11.9.1)

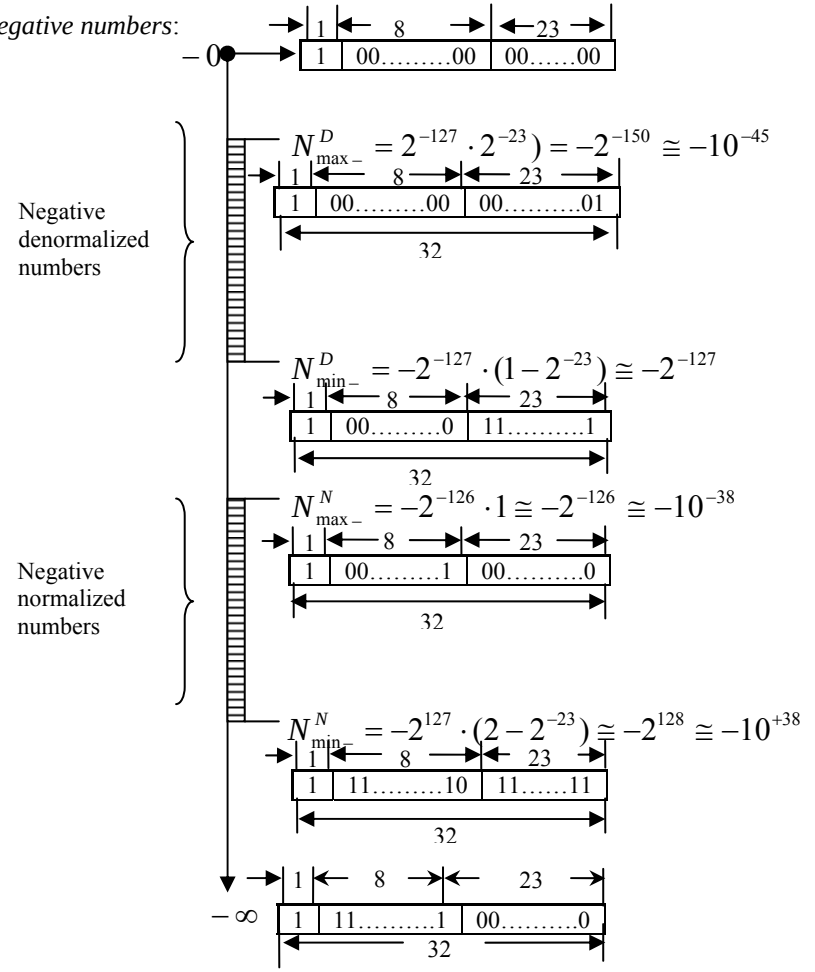
3.11.10 IEEE FLP standard 754-85 representation on real axis

A) Simple Precision (**SP**):

1) *Positive numbers:*



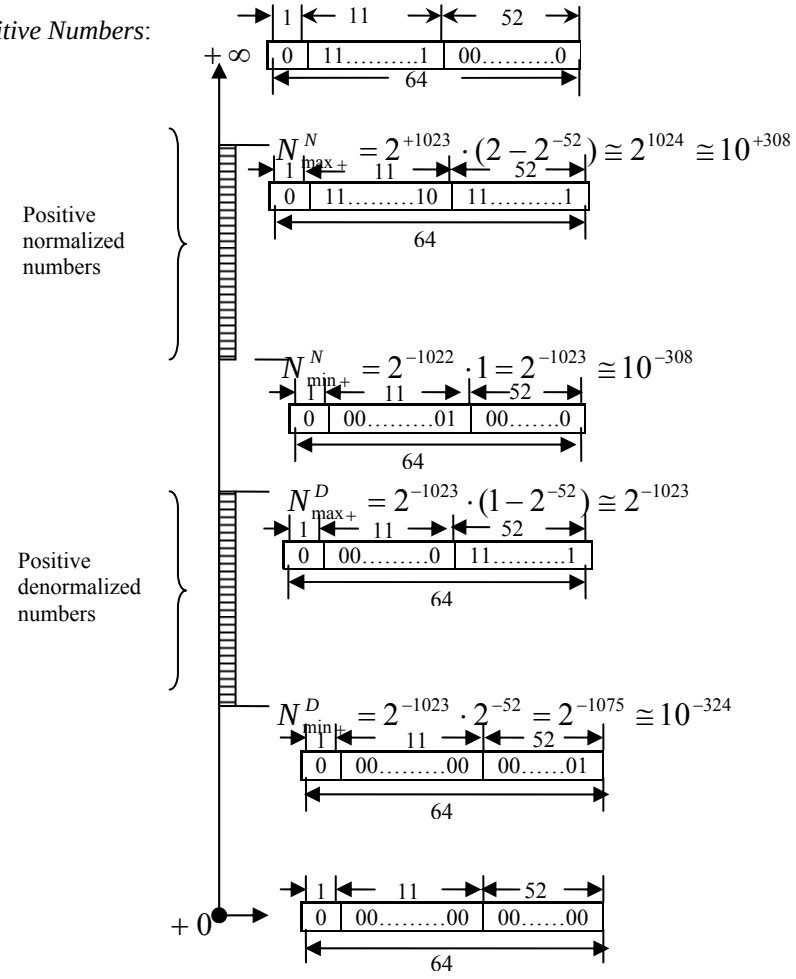
2) *Negative numbers*:



(11.10.2)

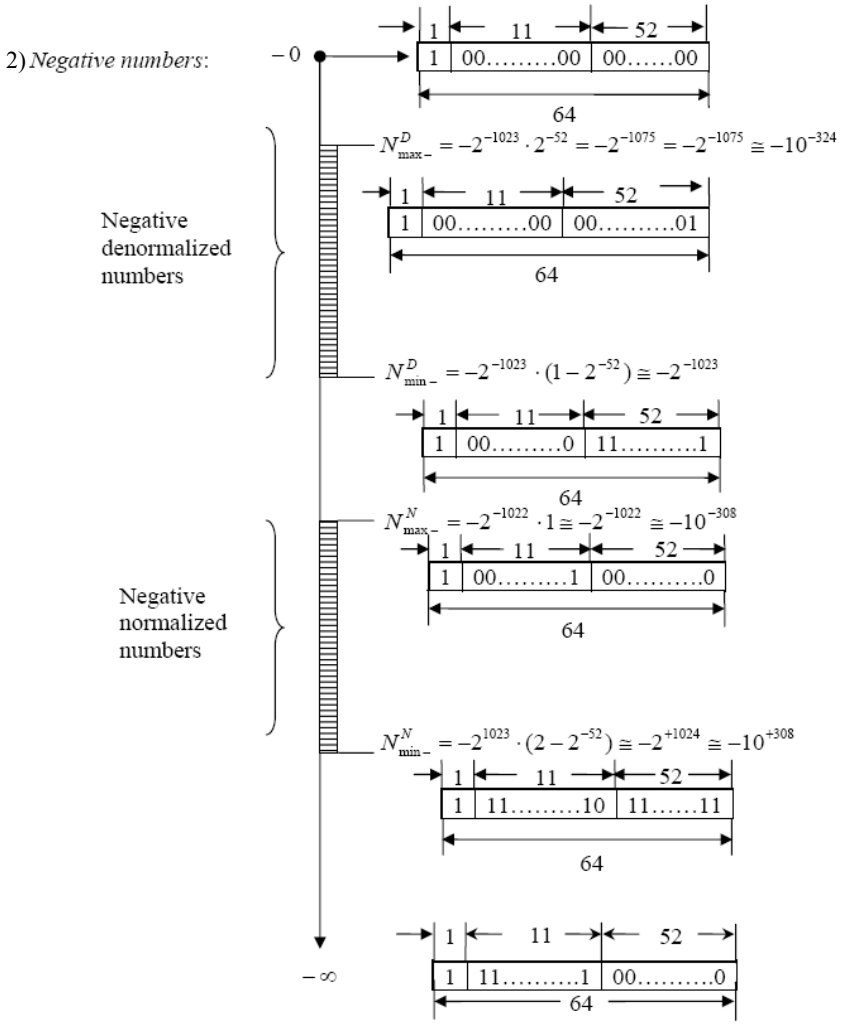
B) Double Precision (**DP**):

1) *Positive Numbers*:



(11.10.3)

2) *Negative numbers*:



(11.10.4)

§3.4 Representation of alphanumeric information

3.4.1 Introduction

- So far, only representation of numbers was considered (4.1.1)
- The information manipulated in a digital computer is wider than numerical values: alphabetic characters (A,B, …,Z) , arithmetic operation symbols (+,−,×,÷), punctuation symbols (, ; . !) , mnemonics for different commands (DEL, ESC, ACK etc) a.s.o. (4.1.2)
- Thus, an extended code is required to encompass all these characters forming the **alphanumeric character set** of a digital computer, known also as the **ALPHABET** of a digital computer. (4.1.3)
- Anyhow these alphanumeric characters are *binary encoded* inside a digital computer (4.1.4)
- Such an alphabet must comprise binary codes on *n* bits for representation of 10 decimal digits, 26 letters of the usual alphabet (upper case and lower case letters), the punctuation symbols, arithmetic operation symbols, special character symbols , etc. (4.1.5)
- If different systems of coding are used then it must be provided a complex coding/decoding process when two digital computers must communicate. (4.1.6)
- Typically, such an alphanumeric set would contain over 100 characters. (4.1.7)
- The key problem is how to standardize representation

3.4.2. Conditions for the Alphanumeric Character Code

1. The *length of the code* must ensure representation of the **entire set** of characters, symbols, commands. (4.2.1)
   - With *n* bits there are encoded 2^n characters; vice versa, if considering *N* characters, then the binary encoding would require $n>[\log_2 N]$ bits.

   - In the beginning era, for such purposes, it was used Baudot code containing only 5 bits, which is completely insufficient nowadays ($2^5 = 32$ characters); this code was specific for older typewriters, allowing printing only uppercase letters. (4.2.2)
   - At present there were adopted *8 bit alphanumeric codes*, ensuring representation of up to 256 characters. (4.2.3)
2. **Convenient correlation** to the dimension of the *Addressable Information Unit* (AIU).
   - Dimension of AIU and memory organization. (4.2.4)
   - Ideally, an alphanumeric code would occupy a location in the main memory, therefore dimension of the code would comply to the dimension of AIU. (4.2.5)
   - At present, an 8 bit combination represents a byte, which is UIA or a division of UIA. Therefore, a recommended length for the alphanumeric code for most computers is 8. (4.2.6)
   - If shorter, then an inefficient mode of memory utilization occurs, by wasting memory cells. (4.2.7)
3. Convenient **correlation to the decimal numbers** representation.
   - The BCD codes are usually on 4 bits (4.2.8)
   - Ideally, it would be to select an *8 bit alphanumeric code to encompass two BCD combinations*. (4.2.9)
   - Hence, in a memory location with 8 bits there can be placed an AIU, an alphanumeric character or exactly two BCD combinations (two decimal digits). (4.2.10)
4. Incorporating some **numerical facilities** – to ensure realization of some specific alphanumeric operations through existing numeric operations. (4.2.11)

- The **weighting principle** of coding the alphanumeric characters, that is for a succession of related characters there is used an ascending set of binary combinations.
  For instance, assuming n=8, for the succession of alphabetic upper case letters, it is used the following sequence of codes:
  $A \rightarrow 01000001 \rightarrow 41h$
  $B \rightarrow 01000010 \rightarrow 42h$
  $C \rightarrow 01000011 \rightarrow 43h$ (4.2.12)
- This condition is very useful for defining particular *manipulations of alphanumeric information*, like, for instance, comparisons between words for creating lists on inventory, employees, payrolls, etc. (4.2.13)
- *Final Conclusion*: it resulted necessity for an **8 bit length alphanumeric code** incorporating weighting principle. (4.2.14)

§4.3. ASCII-8 and EBCDIC codes

4.3.1. ASCII alphanumeric code

- **ASCII** (American Standard Code for Information Interchange) -the most commonly used code in digital computers (4.3.1.1)
- **ASCII** is a 7 bit code used in telecommunications, but it was extended to 8 bits to be adapted for digital computers alphanumeric information representation. (4.3.1.2)
- The **ASCII** code on 7 bits is given below:

Most significant bits b_6,b_5,b_4 / Least significant bits b_3,b_2,b_1,b_0	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	UT	ESC	+	;	K	|	k	{
1100	FI	FS	,	<	L	\	l	;
1101	CR	GS	-	=	M		m	}
1110	SO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	↓	o	DEL

(4.3.1.3)

- This code allows coding of uppercase and lowercase letters, decimal digits, punctuation symbols, mathematical symbols, control characters for peripheral devices etc. (4.3.1.4)
- To change to an 8 bit code it was appended the eighth bit in the most significant position, b_7. (4.3.1.5)
- Usually this extra bit is a parity bit for error detection mechanisms; sometimes it is adopted a “0”. (4.3.1.6)
- Alternatively, use of the 8^{th} bit is defining an alternate character set; with ASCII-7 there are coded 128 alphanumeric characters whereas with an extended ASCII there can be encoded 256 alpha characters, by including also new graphic characters. (4.3.1.7)

- The weighting principle is respected:

$A \rightarrow 01000001$ $\quad a \rightarrow 01100001$

$B \rightarrow 01000010$ $\quad b \rightarrow 01100010$

$C \rightarrow 01000011$ $\quad c \rightarrow 01100011$

etc. $\quad$ *etc.*

$0 \rightarrow 00110000$

$1 \rightarrow 00110001$

$2 \rightarrow 00110010$

etc.

(4.3.1.8)

- Each ASCII combination is represented in a shorthand notation by two hexa digits (h_s h_i):

$$\underbrace{b_7 b_6 b_5 b_4}_{h_S} \underbrace{b_3 b_2 b_1 b_0}_{h_I}$$

For example:

$A \rightarrow \underbrace{0100}\ \underbrace{0001} \rightarrow 41h$

$B \rightarrow \underbrace{0100}\ \underbrace{0010} \rightarrow 42h$

$C \rightarrow \underbrace{0100}\ \underbrace{0011} \rightarrow 43h$

etc.

(4.3.1.9)

4.3.2. EBCDIC alphanumeric code.

- **EBCDIC** (Extended Binary –Coded Decimal Interchange Code) (4.3.2.1)
- **EBCDIC** was developed by IBM, used on IBM mainframes (also in Romanian computers in family FELIX C). (4.3.2.2)
- **EBCDIC** from the beginning was an 8 bit code allowing representation of 256 alphanumeric characters. (4.3.2.3)

- The **EBCDIC** character code shown in hexadecimal is presented below:

00 NUL	20 DS	40 SP	60 –	80	A0	C0 {	E0 \
01 SOH	21 SOS	41	61 /	81 a	A1 ~	C1 A	E1
02 STX	22 FS	42	62	82 b	A2 s	C2 B	E2 S
03 ETX	23	43	63	83 c	A3 t	C3 C	E3 T
04 PF	24 BYP	44	64	84 d	A4 u	C4 D	E4 U
05 HT	25 LF	45	65	85 e	A5 v	C5 E	E5 V
06 LC	26 ETB	46	66	86 f	A6 w	C6 F	E6 W
07 DEL	27 ESC	47	67	87 g	A7 x	C7 G	E7 X
08	28	48	68	88 h	A8 y	C8 H	E8 Y
09	29	49	69	89 i	A9 z	C9 I	E9 Z
0A SMM	2A SM	4A ¢	6A ‘	8A	AA	CA	EA
0B VT	2B CU2	4B	6B ,	8B	AB	CB	EB
0C FF	2C	4C <	6C %	8C	AC	CC	EC
0D CR	2D ENQ	4D (	6D _	8D	AD	CD	ED
0E SO	2E ACK	4E +	6E >	8E	AE	CE	EE
0F SI	2F BEL	4F |	6F ?	8F	AF	CF	EF
10 DLE	30	50 &	70	90	B0	D0 }	F0 0
11 DC1	31	51	71	91 j	B1	D1 J	F1 1
12 DC2	32 SYN	52	72	92 k	B2	D2 K	F2 2
13 TM	33	53	73	93 l	B3	D3 L	F3 3
14 RES	34 PN	54	74	94 m	B4	D4 M	F4 4
15 NL	35 RS	55	75	95 n	B5	D5 N	F5 5
16 BS	36 UC	56	76	96 o	B6	D6 O	F6 6
17 IL	37 EOT	57	77	97 p	B7	D7 P	F7 7
18 CAN	38	58	78	98 q	B8	D8 R	F8 8
19 EM	39	59	79	99 r	B9	D9	F9 9
1A CC	3A	5A !	7A :	9A	BA	DA	FA |
1B CU1	3B CU3	5B $	7B #	9B	BB	DB	FB
1C IFS	3C DC4	5C .	7C @	9C	BC	DC	FC
1D IGS	3D NAK	5D)	7D `	9D	BD	DD	FD
1E IRS	3E	5E ;	7E =	9E	BE	DE	FE
1F IUS	3F SUB	5F ¬	7F “	9F	BF	DF	FF

(4.3.2.4)

- Most of computers would accept alphanumerical data in either code (**ASCII** or **EBCDIC**) and perform conversion to the native code. (4.3.2.5)
- The conversion to ASCII/EBCDIC and back is accomplished in I/O units. (4.3.2.6)
- When a user types in a series of characters at a keyboard by means of an encoding chip that exist inside the keyboard interface, each character is translated into its ASCII or EBCDIC equivalent on 8 bits, followed by transmission of the corresponding byte to CPU. (4.3.2.7)

- The output that a CPU sends back to the display is also an ASCII or EBCDIC code series (alpha information). These codes are translated by peripheral unit interfaces into an understandable information by the user. (4.3.2.8)

§4.4. Alphanumeric information

- *Alphanumeric word* – series of alphanumeric characters. (4.4.1)
- The main characteristic of alphanumeric information representation is its *variable length*. (4.4.2)
- There are upper limits specific for different computer families (for example, 256 characters). (4.4.3)
- Example: "House 1" will be represented in ASCII as follows: (4.4.4)

0010	0010	0100	1000	0100	1111	0101	0101	0101	0011
2	2	4	8	4	F	5	5	5	3
"		H		O		U		S	

0100	0101	0010	0000	0011	0001	0010	0010
4	5	2	0	3	1	2	2
E		b		1		"	

§4.5. Unicode character set

- The **ASCII** and **EBCDIC** codes support the historically dominant (Latin) character sets in computers. But there are many character sets in the world, therefore a new universal character standard was developed that supports a great variety of the world's character sets, called **Unicode**. (4.5.1)
- This is a 16 bit coding system and represents an evolving standard. It changes as new character sets are introduced into it and as existing characters sets evolve and their representation are redefined. (4.5.2)
- Each **Unicode** binary code is represented by a *pattern of 4 hexadecimal digits.* (4.5.3)

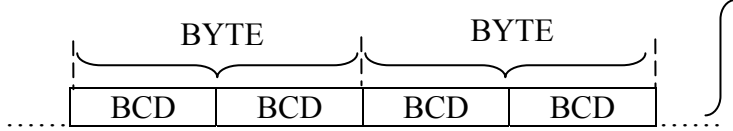
- In the version 2.0 of the **Unicode** standard there are 38885 different coded characters covering the principal written languages of all continents. (4.5.4)
- The first 256 combinations of Unicode, shown in hexadecimal are given in what follows: (4.5.5)

0000	0020 SP	0040 @	0060 `	0080 Ctrl	00A0	00C0 À	00E0 à
NULL	0021 !	0041 A	0061 a	0081 Ctrl	NBS	00C1 Á	00E1 á
0001 SOH	0022 "	0042 B	0062 b	0082 Ctrl	00A1 ¡	00C2 Â	00E2 â
0002 STX	0023 #	0043 C	0063 c	0083 Ctrl	00A2 ¢	00C3 Ã	00E3 ã
0003 ETX	0024 $	0044 D	0064 d	0084 Ctrl	00A3 £	00C4 Ä	00E4 ä
0004 EOT	0025 %	0045 E	0065 e	0085 Ctrl	00A4 ¤	00C5 Å	00E5 å
0005ENQ	0026 &	0046 F	0066 f	0086 Ctrl	00A5 ¥	00C6 Æ	00E6 æ
0006 ACK	0027 '	0047 G	0067 g	0087 Ctrl	00A6 ¦	00C7 Ç	00E7 ç
0007 BEL	0028 (	0048 H	0068 h	0088 Ctrl	00A7 §	00C8 È	00E8 è
0008 BS	0029)	0049 I	0069 i	0089 Ctrl	00A8 ¨	00C9 É	00E9 é
0009 HT	002A *	004A J	006A j	008A Ctrl	00A9 ©	00CA Ê	00EA ê
000A LF	002B +	004B K	006B k	008B Ctrl	00AA ª	00CB Ë	00EB ë
000B VT	002C `	004C L	006C l	008C Ctrl	00AB «	00CC Ì	00EC ì
000C FF	002D -	004D M	006D m	008DCtrl	00AC ¬	00CD Í	00ED í
000D CR	002E .	004E N	006E n	008E Ctrl	00AD -	00CE Î	00EE î
000E SO	002F /	004F O	006F o	008F Ctrl	00AE ®	00CF Ï	00EF ï
000F SI	0030 0	0050 P	0070 p	0090 Ctrl	00AF ¯	00D0 Đ	00F0 ð
0010 DLE	0031 1	0051 Q	0071 q	0091 Ctrl	00B0 °	00D1 Ñ	00F1 ñ
0011 DC1	0032 2	0052 R	0072 r	0092 Ctrl	00B1 ±	00D2 Ò	00F2 ò
0012 DC2	0033 3	0053 S	0073 s	0093 Ctrl	00B2 ²	00D3 Ó	00F3 ó
0013 DC3	0034 4	0054 T	0074 t	0094 Ctrl	00B3 ³	00D4 Ô	00F4 ô
0014 DC4	0035 5	0055 U	0075 u	0095 Ctrl	00B4 ´	00D5 Õ	00F5 õ
0015 NAK	0036 6	0056 V	0076 v	0096 Ctrl	00B5 µ	00D6 Ö	00F6 ö
0016 SYN	0037 7	0057 W	0077 w	0097 Ctrl	00B6 ¶	00D7 ×	00F7 ÷
0017 ETB	0038 8	0058 X	0078 x	0098 Ctrl	00B7 ·	00D8 Ø	00F8 ø
0018CAN	0039 9	0059 Y	0079 y	0099 Ctrl	00B8 ¸	00D9 Ù	00F9 ù
0019 EM	003A :	005A Z	007A z	009A Ctrl	00B9 ¹	00DA Ú	00FA ú
001A SUB	003B ;	005B [	007B {	009B Ctrl	00BA º	00DB Û	00FB û
001B ESC	003C <	005C \	007C |	009C Ctrl	00BB »	00DC Ü	00FC ü
001C FS	003D =	005D]	007D }	009D Ctrl	00BC ¼	00DD Ý	00FD þ
001D GS	003E >	005E ^	007E ~	009E Ctrl	00BD ½	00DE ý	00FE Þ
001E RS	003F ?	005F _	007F	009F Ctrl	00BE ¾	00DF §	00FF ÿ
001F US			DEL		00BF ¿		

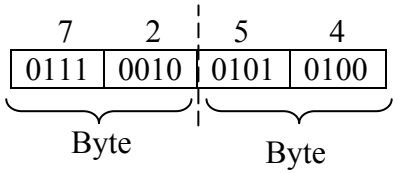
- In defining the **Unicode** there was provided a one to one correspondence between **Unicode** 16 bit pattern and **ASCII** 8 bit patterns, namely between Unicode combinations 0000h up to 007Fh and **ASCII** combinations from 00h up to 7Fh. (4.5.6)
- The 16 bit Unicode standard is *a subset of the 32 bit ISO 10646 Universal Character Set (UCS-4)*. (4.5.7)

§5. Decimal information representation

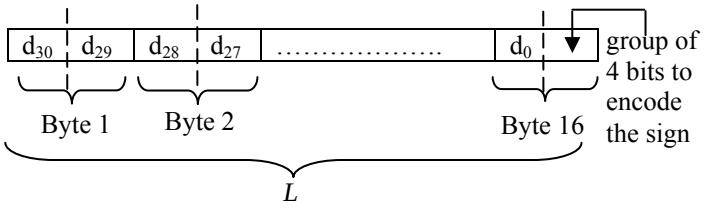
- The modern digital computer can operate on decimal information as well; such information must be coded inside a digital computer through BCD codes. (5.1)
- It is assumed that the mathematical operations are implemented with BCD arithmetic in decimal arithmetical units . (5.2)
- The AIU is a Byte on 8 bits. (5.3)
- Matching of BCD codes to the dimension of an AIU is realized through two formats: **compact** and **zoned**. (5.4)
- **Compact format**: two BCD codes encapsulated in a Byte, as follows: (5.5)



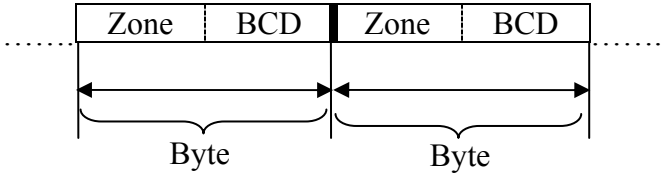
- BCD used is the basic 8421 code. (5.6)
- *Example*: 7254 → in two consecutive Bytes: (5.7)



- The allowed length *L*, of the decimal number, depends on the computer family; for instance, it could be 31 digits + sign: (5.8)



- **Zoned format** assumes each Byte contains a group of 4 bits called **zone**, followed by the BCD code: (5.9)



- **Zone** has different assigned codes depending on the alphanumeric code (ASCII, EBCDIC etc.)
  In EBCDIC → zone=1111
  In ASCII → zone=0011 (5.10)
- Zoned formats are used for Input / Output operations, whereas compact formats are used in processing units. (5.11)
- Automatic *conversions* form zoned to compact formats and vice-versa are design. (5.12)