Proiectare logică

Curs 2

Reprezentarea numerelor. Coduri. Erori

Cristian Vancea

https://users.utcluj.ro/~vcristian/PL.html

Cuprins

- Reprezentarea numerelor în calculator
- Coduri binare
- Detectarea și corectarea erorilor

Reprezentarea numerelor în calculator

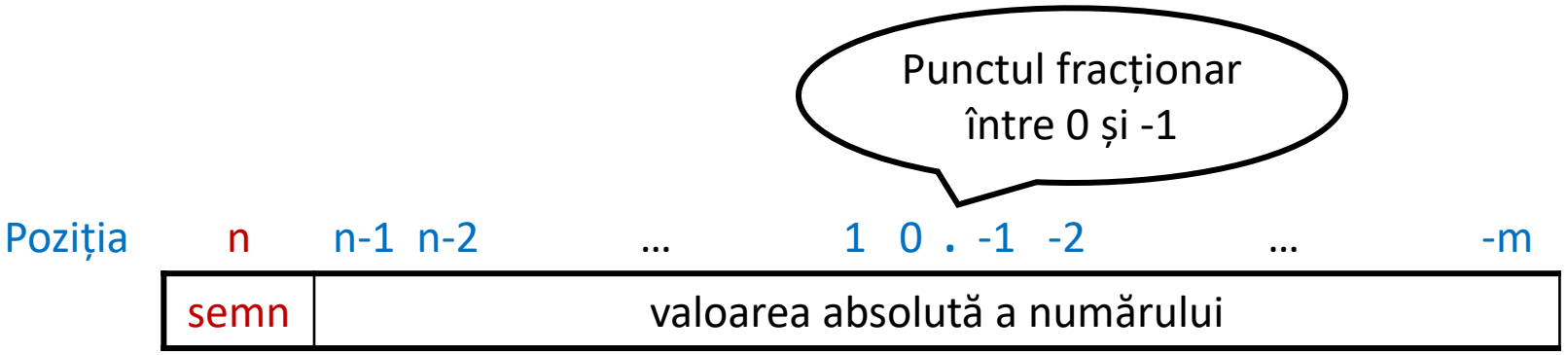
- Are la bază sistemul de numerație **binar**.
- Se utilizează 1 bit pentru semn:
  - bit semn = **0** => număr **pozitiv**
  - bit semn = **1** => număr **negativ**
- Bitul de semn este primul de la stânga (bitul cel mai semnificativ)
- Categorii de reprezentare:
  - Numere întregi
  - Numere fracționare
    - în virgulă fixă
    - în virgulă mobilă

Reprezentarea numerelor în virgulă fixă

- La proiectare se definesc **n** biți parte întreagă și **m** biți partea fracționară => poziția punctului fracționar **nu se schimbă**.
- Nu se consumă biți suplimentari pentru punctul fracționar ci doar se ține cont de poziția lui în calcule.
- Numerele întregi sunt caz particular: **m=0**.
- Variante de reprezentare a semnului:
  - Mărime și semn
  - Complement față de 2

Reprezentarea numerelor în virgulă fixă

Mărime și semn



Ex_1: n=3 m=3

Poziția 3 2 1 0 -1 -2 -3

$0\,1\,1\,0\,.\,1\,0\,1_2 = +(2^2+2^1+2^{-1}+2^{-3})_{10} = +6.625_{10}$

Ex_2: n=4 m=2

Poziția 4 3 2 1 0 -1 -2

$1\,1\,0\,0\,1\,.\,0\,1_2 = -(2^3+2^0+2^{-2})_{10} = -9.25_{10}$

Formula: $N_2 = x_n x_{n-1} \dots x_1 x_0 x_{-1} \dots x_{-m} \Rightarrow N_{10} = (-1)^{x_n} \times \sum_{k=-m}^{n-1} (x_k \times 2^k)$

Reprezentarea numerelor în virgulă fixă

Mărime și semn

Formula: $N_2=x_n x_{n-1} \dots x_1 x_0 x_{-1} \dots x_{-m}$ => $N_{10}=(-1)^{x_n} \times \sum_{k=-m}^{n-1}(x_k \times 2^k)$

Minim: $N_{10}=(-1)^{\mathbf{1}} \times \sum_{k=-m}^{n-1}(\mathbf{1} \times 2^k) = -(2^n - 2^{-m})$

Poziția	n	n-1 n-2	…	1 0 . -1 -2	…	-m
	1	1 1	…	1 1 . 1 1	…	1

Maxim: $N_{10}=(-1)^{\mathbf{0}} \times \sum_{k=-m}^{n-1}(\mathbf{1} \times 2^k) = 2^n - 2^{-m}$

Poziția	n	n-1 n-2	…	1 0 . -1 -2	…	-m
	0	1 1	…	1 1 . 1 1	…	1

Domeniul de reprezentare: $[-(2^n - 2^{-m}), (2^n - 2^{-m})]$

Valoarea 0 acceptă 2 reprezentări: 000…0.00…0 sau 100…0.00…0

Reprezentarea numerelor în virgulă fixă

Mărime și semn – **Adunare**

Ex: n=3 m=1

Transp.	**1 1**					**1 1**
A	0100.1+	=	4.5_{10}+	=	100.1-
B	1001.1	=	-1.5_{10}	=	001.1
R	1110.0	≠	$+3.0_{10}$	=	0011.0

Dezavantaj: Sistemul trebuie să analizeze **biții de semn** pentru a decide operația de efectuat și semnul rezultat conform tabelului.

A_n	B_n	Oper	R_n
0	0	+	0
0	1	-	?
1	0	-	?
1	1	+	1

La scădere se scad valorile absolute, cea mai mică din ce mai mare, iar semnul e dat de numărul cu valoarea absolută cea mai mare.

Reprezentarea numerelor în virgulă fixă

Mărime și semn – **Scădere**

Observație: Scăderea **se poate înlocui cu o adunare** în care se inversează semnul celui de al doilea termen. Deci păstrează dezavantajele adunării.

Ex_1: n=3 m=1

1100.1 **-** **1**001.1 = 1100.1 **+** **0**001.1

inversare semn

Ex_2: n=4 m=2

11110.01 **-** **0**0111.10 = 11110.01 **+** **1**0111.10

inversare semn

Reprezentarea numerelor în virgulă fixă

Complement

Definiție: Complementul lui N_b reprezentat cu $\boldsymbol{n}$ cifre pentru partea întreagă și $\boldsymbol{m}$ cifre pentru partea fracționară este:

$$\boldsymbol{\overline{N}_b = b^n - b^{-m} - N_b}$$

Ex: $n = 3,\ m = 2$

$\overline{78}_{10} = 10^3 - 10^{-2} - 78 = 999.99 - 78 = 921.99_{10}$

$\overline{11.1}_2 = 2^3 - 2^{-2} - 11.1 = 111.11 - 11.1 = 100.01_2$

$n = 2,\ m = 1$

$\overline{23.1}_{10} = 10^2 - 10^{-1} - 23.1 = 99.9 - 23.1 = 76.8_{10}$

$\overline{10.1}_2 = 2^2 - 2^{-1} - 10.1 = 11.1 - 10.1 = 1_2$

de n ori de m ori

Obs: $\overline{N}_{10} = (\mathbf{99\ldots9.99\ldots9})_{\mathbf{10}} - N_{10}$ => scădere din valoarea maximă

$\overline{N}_2 = (\mathbf{11\ldots1.11\ldots1})_{\mathbf{2}} - N_2$ => scădere din valoarea maximă

Reprezentarea numerelor în virgulă fixă

Complement

Ex$_1$: $b = 2,\ n = 3,\ m = 2$

N_2 → 111.11-
$\overline{N}_2$ → 011.10 / 100.01 ↻ inv

N_2 → 111.11-
$\overline{N}_2$ → 101.00 / 010.11 ↻ inv

N_2 → 111.11-
$\overline{N}_2$ → 001.11 / 110.00 ↻ inv

Ex$_2$: $b = 2,\ n = 2,\ m = 4$

N_2 → 11.1111-
$\overline{N}_2$ → 11.0011 / 00.1100 ↻ inv

N_2 → 11.1111-
$\overline{N}_2$ → 01.1011 / 10.0100 ↻ inv

N_2 → 11.1111-
$\overline{N}_2$ → 00.0010 / 11.1101 ↻ inv

Observație: În baza 2 complementul unui număr se poate obține ușor prin **inversarea valorilor biților** din 1 în 0 și din 0 în 1.

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Definiție$_1$: Complementul unui număr binar se mai numește:
complementul față de 1.

Definiție$_2$: **Complementul față de 2** al unui număr binar N_2 cu $\boldsymbol{n}$ biți parte întreagă și $\boldsymbol{m}$ biți de precizie este:

$$\overline{\overline{N}}_2 = 2^n - N_2 = \overline{N}_2 + 2^{-m},$$

adică după inversarea biților **se adună 1 la bitul ce mai puțin semnificativ** (cel mai din dreapta) al complementului față de 1.

Ex$_1$: $n = 3,\ m = 2$

N_2	010.01	101.00
inv		1 1
$\overline{N}_2$	101.10+	010.11+
	000.01 ⇔ 2^{-2}	000.01 ⇔ 2^{-2}
$\overline{\overline{N}}_2$	101.11	011.00

Ex$_2$: $n = 1,\ m = 3$

N_2	1.001	0.011
inv		
$\overline{N}_2$	0.110+	1.100+
	0.001 ⇔ 2^{-3}	0.001 ⇔ 2^{-3}
$\overline{\overline{N}}_2$	0.111	1.101

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Ex_1:	$n = 3,\ m = 2$		Ex_2:	$n = 1,\ m = 3$	
N_2 ⟹	010.01	101.00		1.001	0.011
$\overline{\overline{N}}_2$ ⟹	101.11	011.00		0.111	1.101

Observație: Complementul față de 2 se mai poate obține **inversând biții mai semnificativi decât cel mai puțin semnificativ bit de 1** ⇔ se parcurge numărul de la dreapta la stânga și toți biții următori primului bit de 1 întâlnit se inversează.

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Observație: La reprezentarea pe $\boldsymbol{n}$ biți pentru partea întreagă numărul $\boldsymbol{2^n}$ **are valoarea 0** deoarece se exclude bitul **1**, cel mai semnificativ, fiindcă depășește domeniul de reprezentare.

Ex$_1$: $n = 3,\ m = 2$

$0_{10} = 000.00_2$

$(2^3)_{10} = 1000.00_2$

Ex$_2$: $n = 5,\ m = 1$

$0_{10} = 00000.0_2$

$(2^5)_{10} = 100000.0_2$

bit exclus în reprezentare

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Obs.: Pentru $\boldsymbol{n}$ biți parte întreagă avem: $-\boldsymbol{N_2} = \boldsymbol{0} - \boldsymbol{N_2} = \boldsymbol{2^n} - \boldsymbol{N_2} = \overline{\overline{\boldsymbol{N}}}_{\boldsymbol{2}}$

Regulă$_1$: În reprezentarea complement față de 2, valoarea negativă a unui număr binar este reprezentată de complementul față de 2 al acestuia.

Regulă$_2$: Conversia în baza 10 a lui $N_2 = x_{n-1}x_{n-2} \dots x_1x_0.x_{-1} \dots x_{-m}$ reprezentat în complement față de 2 pe $\boldsymbol{n}$ biți parte întreagă și $\boldsymbol{m}$ biți parte fracționară este:

$$N_{10} = -(x_{n-1} \times 2^{n-1}) + \sum_{k=-m}^{n-2} (x_k \times 2^k)$$

Bitul cel mai semnificativ x_{n-1} indică semnul (0 – pozitiv, 1 – negativ).

Ex$_1$: $n = 3,\ m = 0$

N_{10} ⟹ 3

Poziția **2 1 0**

N_2 ⟹ 0 1 1 ⇔ $2^1+2^0=2+1=3_{10}$

$\overline{\overline{N}}_2$ ⟹ 1 0 1 ⇔ $-2^2+2^0=-4+1=-3_{10}$

Ex$_{1'}$: $n = 3,\ m = 0$

-3

2 1 0

1 0 1 ⇔ $-2^2+2^0=-4+1=-3_{10}$

0 1 1 ⇔ $2^1+2^0=2+1=3_{10}$

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Ex$_2$: $n = 4,\ m = 1$

N_{10} ⟹ 7 . 5

Poziția **3 2 1 0 -1**

N_2 ⟹ 0 1 1 1 . 1 ⇔ $2^2+2^1+2^0+2^{-1}=7.5_{10}$

$\overline{\overline{N}}_2$ ⟹ 1 0 0 0 . 1 ⇔ $-2^3+2^{-1}=-8+0.5=-7.5_{10}$

Ex$_{2'}$: $n = 4,\ m = 1$

-7 . 5

3 2 1 0 -1

1 0 0 0 . 1 ⇔ $-2^3+2^{-1}=-8+0.5=-7.5_{10}$

0 1 1 1 . 1 ⇔ $2^2+2^1+2^0+2^{-1}=7.5_{10}$

Ex$_3$: $n = 5,\ m = 2$

N_{10} ⟹ 1 1

Poziția **4 3 2 1 0 -1 -2**

N_2 ⟹ 0 1 0 1 1 . 0 0 ⇔ $2^3+2^1+2^0=11_{10}$

$\overline{\overline{N}}_2$ ⟹ 1 0 1 0 1 . 0 0 ⇔ $-2^4+2^2+2^0=-11_{10}$

Ex$_{3'}$: $n = 5,\ m = 2$

-1 1

4 3 2 1 0 -1 -2

1 0 1 0 1 . 0 0 ⇔ $-2^4+2^2+2^0=-11_{10}$

0 1 0 1 1 . 0 0 ⇔ $2^3+2^1+2^0=11_{10}$

Reprezentarea numerelor în virgulă fixă

Complement față de 2

Formula: $N_2 = x_{n-1} \dots x_1 x_0 x_{-1} \dots x_{-m} \Rightarrow N_{10} = -(x_{n-1} \times 2^{n-1}) + \sum_{k=-m}^{n-2}(x_k \times 2^k)$

Minim: $N_{10} = -(\mathbf{1} \times 2^{n-1}) + \sum_{k=-m}^{n-2}(\mathbf{0} \times 2^k) = -2^{n-1}$

Poziția	n-1	n-2 n-3	…	1 0 . -1 -2	…	-m
	1	0 0	…	0 0 . 0 0	…	0

Maxim: $N_{10} = -(\mathbf{0} \times 2^{n-1}) + \sum_{k=-m}^{n-2}(\mathbf{1} \times 2^k) = 2^{n-1} - 2^{-m}$

Poziția	n-1	n-2 n-3	…	1 0 . -1 -2	…	-m
	0	1 1	…	1 1 . 1 1	…	1

Domeniul de reprezentare: $[-2^{n-1}, 2^{n-1} - 2^{-m}]$

Reprezentare unică pentru fiecare valoare.

Reprezentarea numerelor în virgulă fixă

Complement față de 2 – **Adunare**

Ex_1: n=4, m=1

A	0111.1+ ⇔ $2^2+2^1+2^0+2^{-1}= 7.5_{10}$		7.5_{10}+
B	1000.0 ⇔ $-2^3= -8_{10}$		-8.0_{10}
R	1111.1 ⇔ $-2^3+2^2+2^1+2^0+2^{-1}= -0.5_{10}$	=	-0.5_{10}

Ex_2: n=4, m=2

Transport 11

A	1111.10+⇔ $-2^3+2^2+2^1+2^0+2^{-1}= -0.5_{10}$		-0.50_{10}+
B	1100.01 ⇔ $-2^3+2^2+2^{-2}= -3.25_{10}$		-3.25_{10}
R	1011.11 ⇔ $-2^3+2^1+2^0+2^{-1}+2^{-2} = -3.75_{10}$	=	-3.75_{10}

se ignoră

Observație: Bitul se semn este utilizat la adunare ca orice alt bit, iar **transportul** generat de bitul de semn **se ignoră**.

Avantaj: Adunarea se simplifică față de reprezentarea în mărime și semn fiindcă nu necesită analiza biților de semn.

Reprezentarea numerelor în virgulă fixă

Complement față de 2 – **Adunare**

- **Depășirea** domeniului de reprezentare: dacă numerele au același semn și rezultatul are semn diferit.

Ex_1: n=4, m=2

Transport 11111

A	0100.10+	$\Leftrightarrow 2^2+2^{-1}= 4.5_{10}$	$4.50_{10}+$
B	0011.11	$\Leftrightarrow 2^1+2^0+2^{-1}+2^{-2} = 3.75_{10}$	3.75_{10}
R	1000.01	$\Leftrightarrow -2^3+2^{-2}= -7.75_{10}$	$\neq 8.25_{10}$

Ex_2: n=4, m=2

Transport 1 1 1

A	1000.11+	$\Leftrightarrow -2^3+2^{-1}+2^{-2} = -7.25_{10}$	$-7.25_{10}+$
B	1110.11	$\Leftrightarrow -2^3+2^2+2^1+2^{-1}+2^{-2}= -1.25_{10}$	-1.25_{10}
R	0111.10	$\Leftrightarrow 2^2+2^1+2^0+2^{-1} = 7.5_{10}$	$\neq -8.50_{10}$

Reprezentarea numerelor în virgulă fixă

Complement față de 2 – **Scădere**

Ex_1: n=4, m=1

Împrumut 1 1 1 1

A 0110.1- $\Leftrightarrow 2^2+2^1+2^{-1}= 6.5_{10}$ 6.5_{10}-

B 0111.0 $\Leftrightarrow 2^2+2^1+2^0 = 7_{10}$ 7.0_{10}

R 1111.1 $\Leftrightarrow -2^3+2^2+2^1+2^0+2^{-1}= -0.5_{10}$ = -0.5_{10}

se ignoră

Ex_2: n=4, m=2

Împrumut 1 1

A 1101.10- $\Leftrightarrow -2^3+2^2+2^0+2^{-1}= -2.5_{10}$ -2.50_{10}-

B 0010.01 $\Leftrightarrow 2^1+2^{-2}= 2.25_{10}$ 2.25_{10}

R 1011.01 $\Leftrightarrow -2^3+2^1+2^0+2^{-2} = -4.75_{10}$ = -4.75_{10}

Observație: Bitul se semn este utilizat la scădere ca orice alt bit, iar **împrumutul** generat de bitul de semn **se ignoră**.

Avantaj: Scăderea se simplifică față de reprezentarea în mărime și semn fiindcă nu necesită analiza biților de semn.

Reprezentarea numerelor în virgulă fixă

Complement față de 2 – **Scădere**

- **Depășirea** domeniului de reprezentare: dacă numerele au semn diferit și rezultatul are același semn cu scăzătorul.

Ex: n=4, m=2

Împrumut 1 1 1 1 1

A 1100.10- $\Leftrightarrow$ $2^{-3}+2^{2}+2^{-1}= -3.5_{10}$ -3.50_{10}-

B 0101.11 $\Leftrightarrow$ $2^{2}+2^{0}+2^{-1}+2^{-2} = 5.75_{10}$ 5.75_{10}

R 0110.11 $\Leftrightarrow$ $2^{2}+2^{1}+2^{-1}+2^{-2}= 6.75_{10}$ $\neq$ -9.25_{10}

- **Observație:** Scăderea este echivalentă cu adunarea cu complementul față de 2 al scăzătorului.

Ex: n=4, m=2

Împrumut 1 1 1 1 **Transport 1 1**

A 1101.10- A 1101.10+

B 0011.11 $\overline{\overline{B}}$ 1100.01

R 1001.11 = R 1001.11

Reprezentarea numerelor în virgulă mobilă

- Se utilizează pentru numere foarte mari sau mici cu grad de precizie ridicat.
- La proiectare se definesc **m** biți pentru **mantisa M** și **e** biți pentru **exponentul E**.
- Reprezentare:

Formula: $N_{10}=(-1)^{SM}\times M \times 2^{(-1)^{SE}\times E}$

- **Exponentul** indică ordinul de mărime printr-o putere.
- **Mantisa** determină valoarea în cadrul ordinului de mărime.

Reprezentarea numerelor în virgulă mobilă

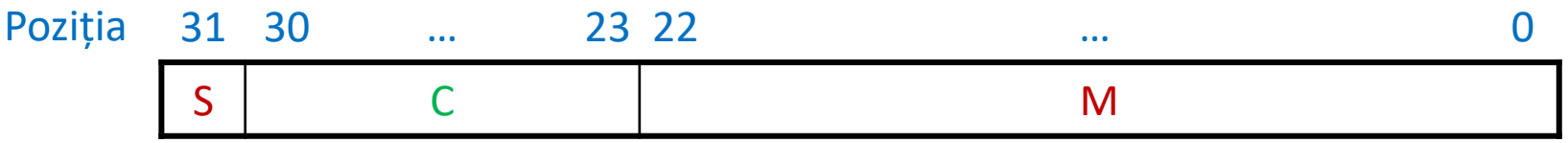
Specificații de implementare

- Se renunță la semnul exponentului **SE** și se introduce mărimea numită **caracteristică (C)**, care are numai valori pozitive:

  C = **E** + deplasament, deplasament > 0

- Se adoptă **forma normalizată** în care partea întreagă a mantisei este 1 și nu se memorează => se câștigă un bit de precizie la mantisă. În acest caz valoarea 0 necesită reprezentare specială.

- Reprezentarea în simplă precizie (32 biți) – IEEE 754



e=8, m=23, deplasament = 127

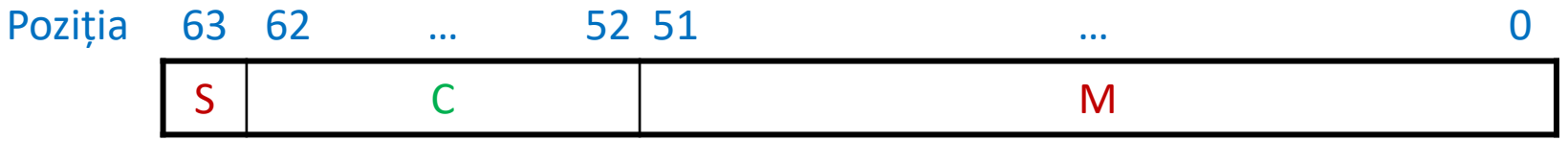
Formula: $N_{10} = (-1)^{S} \times (1.x_{22}x_{21} \ldots x_0)_2 \times 2^{C-127}$, C ϵ [1, 254]

C ϵ [1, 254] ⇔ E ϵ [-126, 127], M ϵ [1, 2 - 2^{-23}]

C = 0 și C = 255 sunt rezervate pentru *numere speciale* (ex: 0, ∞)

Reprezentarea numerelor în virgulă mobilă

- Reprezentarea în dublă precizie (64 biți) – IEEE 754



e=11, m=52, deplasament = 1023

Formula: $N_{10} = (-1)^S \times (1.x_{51}x_{50}\ldots x_0)_2 \times 2^{C-1023}$, C ϵ [1, 2046]

C ϵ [1, 2046] ⇔ E ϵ [-1022, 1023], M ϵ [1, 2 - 2^{-52}]

C = 0 și C = 2047 sunt rezervate pentru *numere speciale* (ex: 0, ∞)

Ex: e=4, m=4, deplasament=7

N_{10} = 0.3125 => N_2 = 0.010100 = 1.0100 x 2^{-2} = +1.0100 x 2^{5-7} => 0 0101 0100

N_{10} = -0.173 => N_2 = -0.0010110 = -1.0110 x 2^{-3} = -1.0110 x 2^{4-7} => 1 0100 0110

N_{10} = -6.25 => N_2 = -110.01 = -1.1001 x 2^2 = -1.1001 x 2^{9-7} => 1 1001 1001

0 1100 1111 => N_2 = +1.1111 x 2^{12-7} = 1.1111 x 2^5 = 111110 => N_{10} = 62

1 0010 1000 => N_2 = -1.1000 x 2^{2-7} = -1.1 x 2^{-5} = -0.000011 => N_{10} = -0.046875

1 0111 1100 => N_2 = -1.1100 x 2^{7-7} = -1.11 x 2^0 = -1.11 => N_{10} = -1.75

Coduri binare

- Se utilizează la codificarea cifrelor zecimale în sistemele de calcul sau transferuri de date.
- Cifrele zecimale (0 … 9) se pot reprezenta pe 4 biți.
- Categorii:
  - ponderate
  - neponderate

Coduri ponderate

- Fiecare poziție binară x_i din codificare are asociată o pondere p_i. Suma ponderilor biților cu valoarea 1 dau valoarea cifrei zecimale asociate codului:

$$\mathrm{N} = x_3p_3 + x_2p_2 + x_1p_1 + x_0p_0$$

Coduri ponderate

Exemple

Cifra zecimală	8421 - BCD				2421				642-3			
	p_3	p_2	p_1	p_0	p_3	p_2	p_1	p_0	p_3	p_2	p_1	p_0
	8	4	2	1	2	4	2	1	6	4	2	-3
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	1
2	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1	1	0	1	1
6	0	1	1	0	1	1	0	0	0	1	1	0
7	0	1	1	1	1	1	0	1	1	1	0	1
8	1	0	0	0	1	1	1	0	1	0	1	0
9	1	0	0	1	1	1	1	1	1	1	1	1

- Codul 8421 - BCD (Binary Coded Decimal) ⇔ conversia în baza 2 a fiecărei cifre zecimale.
- Codurile 2421, 642-3 sunt coduri **autocomplementare** fiindcă suma ponderilor = 9.
- La 2421 (cod Aiken) codificările pentru 0-4 trebuie să conțină 0 pe bitul cel mai semnificativ.

Cifrele complementare și codurile asociate lor sunt trecute pe rânduri de culoare identică.

Cod **autocomplementar** – dacă o cifră zecimală A este complementul cifrei B (A = 9 - B) atunci codul binar al cifrei A este complementul codului cifrei B și se obține prin **inversarea biților** din 1 în 0 și din 0 în 1.

Coduri neponderate

Codul Excess 3

Cifra zecimală	8421 - BCD	Exces 3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

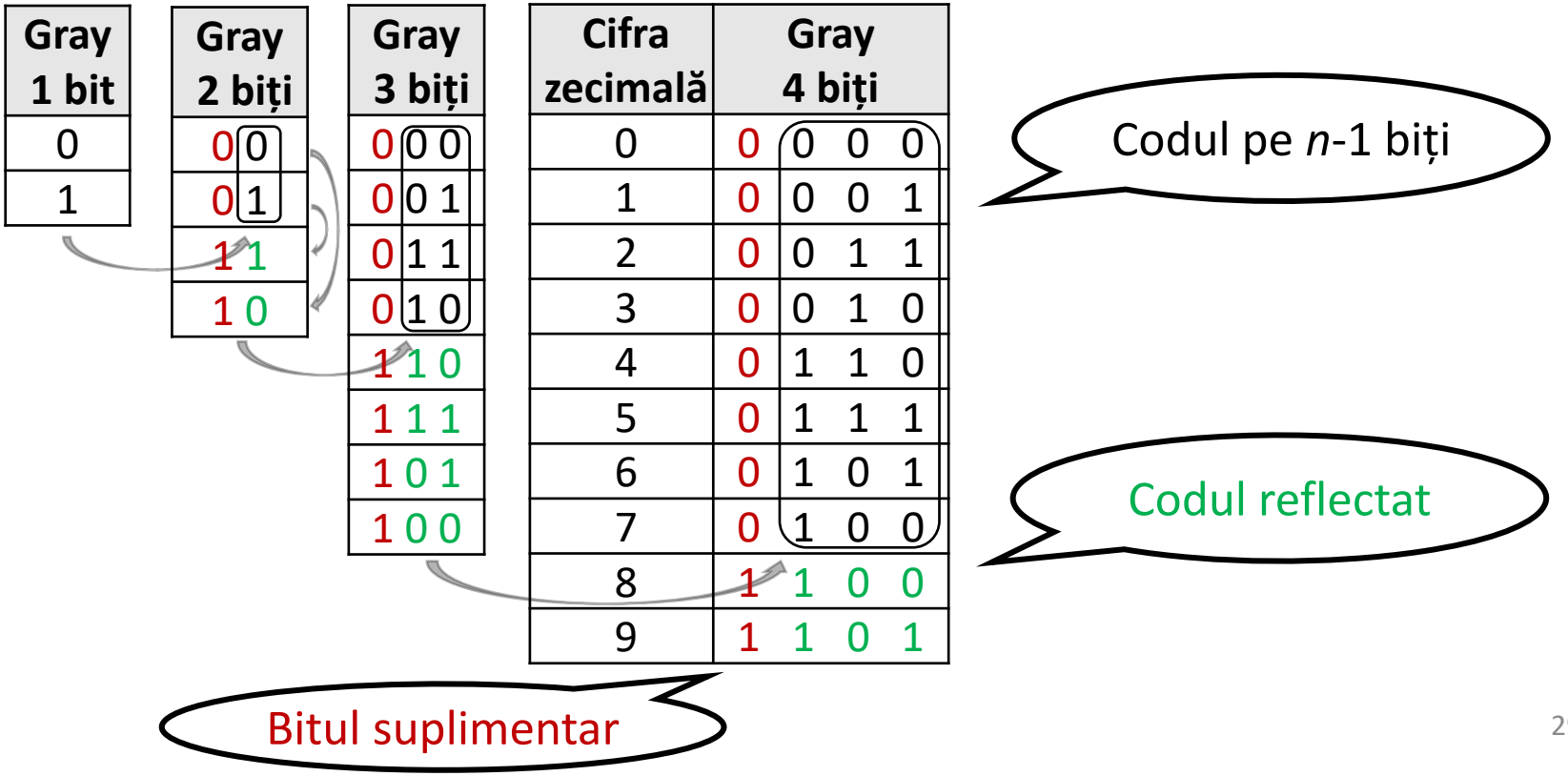
- Codul Excess 3 se obține din BCD prin adunarea valorii 3 (0011 în binar).
- Este un cod **autocomplementar.**

Cifrele complementare și codurile asociate lor sunt trecute pe rânduri de culoare identică.

Coduri neponderate

Codul Gray

- cod **reflectat** – codul pe ***n*** biți se generează prin reflectarea codului pe ***n***-1 biți și adăugarea unui bit suplimentar pe poziția cea mai semnificativă (la stânga): 0 la codul normal și 1 la cel reflectat.
- cod **ciclic** fiindcă oricare **2 coduri consecutive diferă printr-un bit**.



Detectarea și corectarea erorilor

- În procesul de transmitere a informației pot sa apară erori în cadrul datelor transmise.
- S-au adoptat metode de codificare capabile sa detecteze erorile și eventual să le corecteze.
- Dacă nu se poate realiza corectarea la destinație atunci se va recurge la retransmiterea informației afectate => consum suplimentar de timp.

Detectarea erorilor

Metoda bitului de paritate

- Se poate aplica peste orice codificare utilizată.
- La codul utilizat se adaugă un **bit de paritate** în poziția cea mai semnificativă (la stânga):
  - **Paritate pară** – bitul de paritate va avea valoarea 0 sau 1 astfel încât numărul total de biți de 1 să fie par.

    Ex: pentru codul 0110 se adaugă 0 => 0 0110

    pentru codul 0010 se adaugă 1 => 1 0010

    pentru codul 1101 se adaugă 1 => 1 1101
  - **Paritate impară** – bitul de paritate va avea valoarea 0 sau 1 astfel încât numărul total de biți de 1 să fie impar.

    Ex: pentru codul 0110 se adaugă 1 => 1 0110

    pentru codul 0010 se adaugă 0 => 0 0010

    pentru codul 1101 se adaugă 0 => 0 1101
- **Obs**: Detectează erori survenite pe maxim 1 bit.
- **Notă**: Sursa și destinația folosesc aceeași paritate.

Detectarea erorilor

Metoda bitului de paritate

Ex: Să se determine valoarea transmisă în cod Excess 3 de un sistem care folosește paritatea pară pentru următoarele valori zecimale:

Valoare zecimală	Valoare transmisă
98	111001011
36	001101001
56	110001001
73	010100110
102	1010000110101