2 SISTEME DE NUMERAȚIE

2.1 Sisteme de numerație poziționale

Toate calculele pe care le facem în viața de zi cu zi sunt efectuate în sistemul de numerație zecimal, sau în baza 10. Ne-am obişnuit cu acest sistem încă de la naştere, natura fiind cea care a avut rolul decisiv în această alegere. Ea a stabilit ca rezultat final al evoluției un optim de cinci degete la fiecare dintre cele două mâini ale omului. Acest sistem de numerație este un **sistem pozițional**, pentru că orice număr este reprezentat printr-un şir de cifre zecimale, adică cifrele de la 0 la 9, fiecare poziție a cifrei în număr având o anumită **pondere**. Valoarea numărului este o sumă ponderată a cifrelor din care este format numărul.

Exemplul 2.1

Numărul întreg 1734 se scrie sub forma:

$$1734 = 1 \cdot 10^3 + 7 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0,$$

unde 1 este cifra miilor, 7 este cifra sutelor, 3 a zecilor, iar 4 a unităților.

În mod asemănător se reprezintă şi numerele fracționare. Cifrele de la dreapta virgulei vor avea ponderi corespunzătoare, date de exponenții negativi ai puterilor lui 10.

Exemplul 2.2

Numărul 5837,412 se scrie sub forma:

$$5837{,}412 = 5 \cdot 10^3 + 8 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 + 4 \cdot 10^{-1} + 1 \cdot 10^{-2} + 2 \cdot 10^{-3},$$

unde 4 este cifra zecimilor, 1 a sutimilor şi aşa mai departe.

În cazul general, un număr oarecare scris în baza b sub forma:

$$x_{n-1}x_{n-2}\ldots x_1x_0, x_{-1}x_{-2}\ldots x_{-m}\ ,$$

unde $x_i \in \{0,1,\ldots,b-1\}$, are ca valoare suma fiecărei cifre multiplicate cu puterea corespunzătoare a bazei:

$$D = \sum_{i=-m}^{n-1} x_i \cdot b^i$$

Dacă $b = 2$, atunci se obține **reprezentarea binară** a numărului. În sistemele numerice toate numerele se reprezintă în baza 2, deoarece semnalele disponibile au numai două stări, care sunt asociate cifrelor binare 0 şi 1.

Exemplul 2.3

Valoarea în baza 10 a numărului 1101,001 scris în baza 2 este:

$$1101{,}001_2 = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} = 13{,}125_{10}$$

Dacă virgula lipseşte, atunci se presupune că ea se află la dreapta cifrei de pondere minimă. Cifra din stânga reprezentării binare a unui număr este cifra de pondere maximă, sau **bitul cel mai semnificativ** (*MSB - Most Significant Bit*), iar cifra din dreapta este cifra de pondere minimă, sau **bitul cel mai puțin semnificativ** (*LSB - Least Significant Bit*).

Exemplul 2.4

Valoarea în baza 10 a numărului întreg 110100111 scris în baza 2 este:

$$110100111_2 = 1\cdot 2^8 + 1\cdot 2^7 + 0\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 423_{10}$$

Poate că un neajuns al reprezentării binare este numărul mare de biți folosit la scrierea unui număr. Pentru o scriere sub o formă mai compactă se utilizează uneori **reprezentarea hexazecimală**, sau în baza 16, şi mai rar **reprezentarea octală**, sau în baza 8. Aceste baze sunt puteri ale lui 2 şi orice şir de 3 biți poate fi reprezentat în mod unic printr-o cifră octală (de la 0 la 7), după cum orice şir de 4 biți poate fi reprezentat printr-o cifră hexazecimală (cifrele de la 0 la 9 şi simbolurile literale de la *A* la *F* pentru suplinirea celor 6 cifre care nu au echivalent zecimal).

Exemplul 2.5

Numărul binar 11101,1011101 se poate scrie sub una din formele:

$$11101{,}1011101_2 = 011\ 101{,}101\ 110\ 100_2 = 35{,}564_8$$

$$11101{,}1011101_2 = 0001\ 1101{,}1011\ 1010_2 = 1\mathrm{D{,}BA}_{16}$$

Am văzut până acum că anumite grupări convenabile de biți şi **substituția** lor prin cifrele unuia dintre sistemele de numerație în baza 8 sau 16 permit conversia binar-octal sau binar-hexazecimal şi invers. Conversia în sistemul de numerație zecimal se face prin **sumarea** ponderată a cifrelor din care este format numărul. Cum se poate face însă conversia unui număr din sistemul de numerație zecimal într-un alt sistem de numerație?

Un număr întreg scris în baza *b* sub forma $x_{n-1}x_{n-2}\ldots x_1x_0$ are valoarea:

$$D = \sum_{i=0}^{n-1} x_i \cdot b^i = ((\ldots(x_{n-1}\cdot b + x_{n-2})\cdot b + \ldots + x_2)\cdot b + x_1)\cdot b + x_0$$

Observăm acum că prin împărțirea numărului *D* la baza *b* se obține un cât *Q* și un rest x_0. Câtul obținut are același aspect cu formula inițială. Prin împărțirea lui la *b* se obține un nou cât *Q'* şi restul x_1. Cu fiecare nouă împărțire se obține câte o cifră a numărului căutat. Procedura se oprește atunci când câtul devine 0. Restul obținut este cifra cea mai semnificativă din reprezentarea numărului în baza *b*, adică x_{n-1}.

Exemplul 2.6

Pentru a afla reprezentarea binară a numărului zecimal 179, procedăm astfel:

179 : 2 = 89 + **1** (LSB)
89 : 2 = 44 + **1**
44 : 2 = 22 + **0**
22 : 2 = 11 + **0**
11 : 2 = 5 + **1**
5 : 2 = 2 + **1**
2 : 2 = 1 + **0**
1 : 2 = 0 + **1** (MSB)

Așezând în ordine cifrele obținute ca rest, rezultă: $179_{10} = 10110011_2$.

Cum vom proceda însă dacă numărul zecimal are și o parte fracționară? Atunci vom separa partea întreagă de partea fracționară și vom face cele două conversii separat. Observăm că partea fracționară a numărului se poate scrie sub forma:

$$D_F = x_{-1} \cdot b^{-1} + x_{-2} \cdot b^{-2} + \ldots + x_{-m} \cdot b^{-m}$$

Prin multiplicarea cu *b* a membrilor ecuației de mai sus obținem:

$$b \cdot D_F = x_{-1} + x_{-2} \cdot b^{-1} + \ldots + x_{-m} \cdot b^{-m+1}$$

Se poate ușor observa că partea întreagă a expresiei din membrul drept este x_{-1}. Prin scăderea acestei valori și o nouă multiplicare cu *b* se obține coeficientul x_{-2}, ca parte întreagă a expresiei rezultate în membrul drept:

$$b \cdot (b \cdot D_F - x_{-1}) = x_{-2} + x_{-3} \cdot b^{-1} + \ldots + x_{-m} \cdot b^{-m+2}$$

Se continuă aceste calcule atât timp cât multiplicăm cu *b* numere diferite de zero. Este posibil ca în unele situații să nu terminăm niciodată și din acest motiv, un alt criteriu de oprire este precizia acceptată a reprezentării.

Exemplul 2.7

Ne propunem să aflăm reprezentarea binară a numărului zecimal 0,61:

$0,61 \times 2 = 1,22$ **1** (MSB)
$0,22 \times 2 = 0,44$ **0**
$0,44 \times 2 = 0,88$ **0**
$0,88 \times 2 = 1,76$ **1**
$0,76 \times 2 = 1,52$ **1**
$0,52 \times 2 = 1,04$ **1**
$0,04 \times 2 = 0,08$ **0** … și așa mai departe.

Dacă ne oprim aici, obținem un rezultat aproximativ datorită trunchierii $0,61_{10} \approx 0,1001110\ldots_2$.

2.2 Operații cu numere binare

Operațiile aritmetice cu numere reprezentate în baza 2 nu au, din punct de vedere teoretic, caracteristici cu totul speciale. Folosind regulile de operare pe un singur bit şi ținând seamă de capacitatea de reprezentare a operanzilor, se pot face uşor operații cu numere întregi şi pozitive.

Exemplul 2.8

Un exemplu de adunare zecimală şi binară a operanzilor 173_{10} şi 42_{10}:

$$\begin{array}{r} 173\,+ \\ 42 \\ \hline 215 \end{array} \qquad \begin{array}{r} 1\ \ \ 1\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \\ 1\,0\,1\,0\,1\,1\,0\,1\,+ \\ 0\,0\,1\,0\,1\,0\,1\,0 \\ \hline 1\,1\,0\,1\,0\,1\,1\,1 \end{array}$$

Se adună cifrele de la dreapta la stânga, începând cu cele mai puțin semnificative şi se ține seamă de **transportul** care apare când suma cifrelor depăşeşte baza de numerație folosită.

Operația de scădere se poate face asemănător. Pentru un sistem numeric este însă mult mai convenabil să se utilizeze o operație echivalentă, numită **adunarea cu complementul față de doi**. Este mai uşor de efectuat o adunare la care trebuie să ținem seamă de **transport (carry)**, decât o scădere la care trebuie să ținem seamă de **împrumut (borrow)** de la una din cifrele de rang superior.

Complementul unui număr de *n* cifre scris în baza *b*, față de numărul *b*, se obține prin scăderea numărului din b^n. Dacă notăm numărul cu *D*, atunci complementul lui față de *b* este $C = b^n - D$. Operația de scădere poate fi însă evitată dacă scriem rezultatul sub forma $C = ((b^n - 1) - D) + 1$. Dacă definim complementul unei cifre *d* sub forma $c = b - 1 - d$, atunci expresia $(b^n - 1) - D$ se obține prin complementarea cifrelor lui *D*. Deci complementul numărului *D* se poate obține prin complementarea separată a cifrelor numărului *D* şi adunarea cifrei 1, conform relației de mai sus.

Exemplul 2.9

Complementul față de 10 al numărului 1849 este $10^4 - 1849$, adică 8151. Fără a efectua scăderea, observăm că prin complementarea separată a cifrelor 1, 8, 4 şi 9 se obțin cifrele 8, 1, 5 şi respectiv 0. Rezultatul este 8150 + 1 = 8151.

Complementul față de 2 al numărului 119_{10} se calculează în felul următor: $119_{10} = 01110111_2$. Prin complementarea fiecărui bit din această reprezentare se obține 10001000_2, iar prin adunarea lui 1 se obține $10001001_2 = -119_{10}$.

Prin complementarea față de doi se realizează, de fapt, schimbarea semnului unui număr, indiferent de valoarea lui inițială. În reprezentarea binară a numărului 119_{10} s-a introdus pe poziția cea mai semnificativă un bit suplimentar, numit **bit de semn**, care este 0 pentru numere pozitive şi 1 pentru cele negative.

Pentru adunarea numerelor reprezentate în complement față de doi se reprezintă operanzii pe un număr fixat de biți, bitul cel mai semnificativ este bitul de semn, iar eventualul transport obținut prin adunare se ignoră.

Exemplul 2.10

Folosim o reprezentare pe 8 biți, cu bitul cel mai semnificativ ca bit de semn. Numerele întregi care se pot astfel reprezenta sunt cuprinse în intervalul [-127, +127].

25 +	00011001 +	25 +	00011001 +
42	00101010	- 12	11110100
67	01000011	13	1 00001101
25 +	00011001 +	- 17 +	11101111 +
- 30	11100010	- 95	10100001
- 5	0 11111011	- 112	1 10010000

În toate aceste exemple nu s-a depăşit capacitatea de reprezentare de 8 biți.

O dificultate suplimentară apare la detectarea depăşirii capacității de reprezentare. Detectarea **depăşirii (overflow)** se face uşor dacă observăm că ea apare atunci când operanzii au acelaşi semn şi rezultatul are semn contrar.

Exemplul 2.11

Folosim tot o reprezentare pe 8 biți, cu bitul cel mai semnificativ ca bit de semn. Depăşirea apare pentru orice rezultat aflat în afara intervalului [-127, +127].

79 +	01001111 +	- 100 +	10011100 +
56	00111000	- 85	10101011
135	10000111 = - 121	- 185	1 01000111 = + 71

Operațiile de înmulțire sau împărțire se pot face prin adunări sau scăderi repetate. Dacă se foloseşte reprezentarea în complement față de doi, trebuie să avem în vedere un număr suficient de biți pentru reprezentarea operanzilor. Produsul a două numere de *i* biți poate fi reprezentat printr-un număr de cel mult 2*i* biți.

Exemplul 2.12

Produsul numerelor 11 şi 13 este $11 \times 13 = 143$. Numerele binare s-au reprezentat fără semn. Exemplul din dreapta ilustrează produsul numerelor -5 şi -3, reprezentate în complement față de doi. Operanzii sunt reprezentați pe 6 biți, iar produsul lor, fără semn, este format din cei 6 biți mai puțin semnificativi ai rezultatului înmulțirii:

$$\begin{array}{r} 1011 \times \\ \underline{1101} \\ 1011 \\ 0000 \\ 1011 \\ \underline{1011} \\ 10001111 \end{array} = 143 \qquad \begin{array}{r} 111011 \times \\ \underline{111101} \\ 111011 \\ 111011 \\ \cdots\cdots \\ \underline{111011} \\ 111000\underbrace{001111} \end{array} = 15$$

Prin împărțirea lui 29 la 5 se obține câtul 5 şi restul 4:

$$
\begin{array}{r|l}
11101 & 101 \\
\underline{101} & 101 \\
01001 & \\
\underline{101} & \\
100 &
\end{array}
$$

Se observă că înmulțirea unui întreg pozitiv cu 2^i se face prin deplasarea întregii configurații binare cu *i* poziții spre stânga şi introducând bitul 0 pe cele *i* poziții mai puțin semnificative rămase. Pentru împărțirea cu 2^i se face deplasarea spre dreapta cu *i* poziții.

2.3 Alte reprezentări binare

În afară de reprezentarea în complement față de doi, numerele întregi se mai pot reprezenta prin **mărime şi semn**. Pe poziția cea mai semnificativă a reprezentării se adaugă bitul de semn, care prin convenție este 0 pentru numerele pozitive şi 1 pentru cele negative, iar biții următori sunt cei rezultați din conversia numărului fără semn din zecimal în binar.

Exemplul 2.13

Reprezentarea numerelor 153 şi -153 prin mărime şi semn este următoarea:

+153 = 010011001 -153 = 110011001

La fel ca în reprezentarea prin complement față de doi, semnul numărului este dat de valoarea celui mai semnificativ bit. Totuşi această reprezentare este mai comodă pentru noi, deoarece numărul şi semnul sunt reprezentate prin două câmpuri separate.

Există unele aplicații în care se preferă afişarea rezultatelor unor operații binare prin cifre zecimale, mai uşor de interpretat pentru operatorul uman (calculatoare de buzunar). Cele 10 cifre zecimale de la 0 la 9 se înlocuiesc prin numerele binare corespondente, de la 0000 la 1001, rezultate prin conversia lor în binar fără semn. Celelalte 6 combinații binare care pot fi generate cu 4 biți, de la 1010 la 1111, nu sunt folosite. Această codificare este cunoscută sub numele de **cod BCD (<u>B</u>inary-<u>C</u>oded <u>D</u>ecimal)**, sau **cod zecimal codificat binar**. Se poate observa că numărul de biți în reprezentarea BCD este mai mare decât cel folosit în reprezentarea binară.

Exemplul 2.14

Reprezentarea în cod BCD a numărului 153 se face pe 12 biți:

0001 0101 0011

Reprezentarea **binar-zecimală cu exces trei** este o formă particulară de reprezentare BCD, în care fiecare digit este reprezentat prin forma binară a valorii sale sumate cu 3.

Exemplul 2.15

Forma binar-zecimală cu exces trei a numărului 153 este următoarea:

0100 1000 0110

Reprezentarea cu exces trei permite calcularea simplă a complementului față de 9, foarte util în realizarea operațiilor aritmetice. Complementul față de 9 al unui număr în exces trei se obține prin complementarea bit cu bit (complementul față de unu).

Exemplul 2.16

Complementul față de 9 al unui număr zecimal este complementul față de 10 din care se scade o unitate. Complementul numărului 153 în exces trei este:

1011 0111 1001,

care reprezintă, într-adevăr, numărul 846.

Reprezentarea **1 prin *m*** presupune reprezentarea a *m* numere, fiecare număr având un cod binar pe *m* biți. Un singur bit are valoarea 1, toți ceilalți *m - 1* biți fiind 0. Diferența dintre numere este dată de poziția bitului de valoare 1 în secvența celor *m* biți.

Exemplul 2.17

Patru stări ale unui sistem numeric pot fi codificate folosind codul 1 prin 4. Reprezentarea binară a stărilor este: 1000, 0100, 0010 şi 0001.

Această codificare binară a stărilor poate fi deosebit de utilă pentru sinteza unor structuri numerice iterative (adică structuri care se repetă de mai multe ori într-un sistem).

Trecerea de la un număr binar la altul imediat următor se poate face prin modificarea unui număr mai mare sau mai mic de biți. Astfel, de la 2 la 3 comută un singur bit (0010 → 0011), dar de la 7 la 8 comută patru biți (0111 → 1000). Există situații în care este utilă o reprezentare binară în care tranziția între numere succesive să fie făcută cu un număr minim de comutări de biți. Reprezentarea numerelor binare în **cod Gray** alocă numerelor succesive coduri care diferă printr-un singur bit. Se spune că într-o secvență de numărare Gray **codurile sunt adiacente**.

Exemplul 2.18

Conversia din cod binar în cod Gray, într-o reprezentare pe 3 biți, este dată de corespondențele următoare:

000 → 000	100 → 110
001 → 001	101 → 111
010 → 011	110 → 101
011 → 010	111 → 100

Regula de generare a codului Gray este foarte simplă: bitul de rang *i* din reprezentarea în cod Gray este 0 dacă biții de rang *i* şi *i*+1 din reprezentarea binară corespondentă sunt egali; în caz contrar, bitul respectiv are valoarea logică 1. Bitul de la stânga celui mai semnificativ bit din reprezentarea binară este considerat 0.

Reprezentarea în virgulă mobilă se face pentru aproximarea numerelor reale. O reprezentare cu mare precizie a unui număr real presupune utilizarea unui număr foarte mare de cifre binare. Pentru a evita acest lucru, numerele se scriu sub forma:

$$N = \pm 0,M \times 2^{E},$$

unde M şi E sunt configurații binare. Mărimea subunitară 0,M se numeşte **mantisa** numărului, iar E este un întreg care reprezintă **exponentul** acestuia. Pentru semnul numărului se foloseşte un bit separat.

Exemplul 2.19

Pentru M = 11001 şi E = 100 numărul reprezentat este 12,5. Observăm că:

$$0{,}11001 \times 2^{100} = 1100{,}1$$

Refaceți calculul zecimal pentru operația $0{,}78125 \times 2^4$ şi obțineți acelaşi rezultat. Se observă şi aici că înmulțirea numărului de 4 ori cu 2 s-a făcut prin deplasarea virgulei spre dreapta cu 4 poziții.

Această reprezentare poate fi folosită în egală măsură pentru numere foarte mari şi pentru numere foarte mici, datorită exponentului E. Ea asigură o precizie relativă dată de numărul de biți cu care este reprezentată mantisa M.

Reprezentarea în virgulă mobilă a fost **standardizată** (IEEE Standard 754). Standardul precizează dimensiunea mantisei, a exponentului, modul de codificare şi poziția relativă în reprezentare. **Formatul simplu** al standardului prevede o codificare pe 32 biți: primul bit de pe poziția cea mai semnificativă este bitul de semn, următorii 8 biți reprezintă exponentul în exces 127, iar ceilalți 23 biți rămaşi reprezintă cei 23 biți mai puțin semnificativi ai mantisei. Bitul cel mai semnificativ al mantisei este întotdeauna 1 şi din acest motiv el lipseşte din formatul reprezentării.

Exemplul 2.20

Cuvântul de 32 biți de mai jos reprezintă conform standardului IEEE 754 un număr în virgulă mobilă:

0 10001100 00110010000000000000000

Numărul este pozitiv, exponentul este 140 - 127 = 13, iar mantisa se scrie pe 24 biți: 100110010000000000000000. Prin mutarea virgulei cu 13 poziții spre dreapta se obține 1001100100000, adică reprezentarea binară a numărului 4896.

Standardul prevede şi un **format dublu** care foloseşte o reprezentare pe 64 biți.

Probleme

2.1 Realizați conversia în baza 10 a următoarelor numere: 1101011_2 ; 174003_8 ; 1201_3 ; $63{,}52_8$; $AB3D_{16}$; $C79{,}FE_{16}$.

2.2 Realizați conversia în baza 8 a următoarelor numere zecimale: 16 ; 365 ; 3489 ; 97,12.

2.3 Realizați conversia în baza 16 a următoarelor numere zecimale: 61453 ; 23851 ; 5719; 23,25; 574,32 .

2.4 Realizați conversia în baza 2 a numerelor zecimale 2,71 şi 3,14. Se foloseşte o reprezentare pe 8 biți. Estimați care este eroarea de reprezentare şi arătați dacă este acceptabilă pentru numărul de biți ai reprezentării.

2.5 Realizați conversia numărului zecimal 100 în toate bazele de la 2 la 16.

2.6 Fiecare din următoarele operații aritmetice este corectă în cel puțin un sistem de numerație. Stabiliți bazele posibile pentru fiecare din aceste operații:

a) 1234 + 5432 = 6666 *b*) 41/3 = 13
c) 33/3 = 11 *d*) 23 + 44 + 14 + 32 = 223
e) 302/20 = 12,1 *f*) $\sqrt{41} = 5$

([Wakerly, 1990])

2.7 Prima expediție pe Marte a descoperit numai ruinele unei civilizații. Prin studiul obiectelor găsite, exploratorii au ajuns la concluzia că ființele respective aveau patru picioare şi un tentacul care se ramifica la capăt cu un număr de "degete". După studii îndelungate, exploratorii au reuşit să traducă matematica marțiană. Ei au descoperit următoarea ecuație:

$$5x^2 - 50x + 125 = 0$$

cu soluțiile date x = 5 şi x = 8. Valoarea x = 5 pare a fi destul de legitimă, în timp ce soluția x = 8 necesită explicații suplimentare. Reflectând la modul în care s-a dezvoltat sistemul de numerație pe Pământ, exploratorii au ajuns la concluzia că sistemul marțian a avut o istorie similară. Câte degete credeți că aveau marțienii?

([Wakerly, 1990])

2.8 Realizați următoarele operații aritmetice, folosind reprezentarea numerelor în complement față de doi:

a) 17 + 5 *b*) - 14 - 12
c) 12 - 6 *d*) 39×18
e) -17×52 *f*) $39 \div (-6)$

2.9 Să presupunem că un număr binar *B* de 4*n* biți este reprezentat de un număr hexazecimal *H* de *n* cifre. Demonstrați că complementul față de 2 al numărului *B* este reprezentat de complementul față de 16 al numărului *H*. Formulați și demonstrați o propoziție similară și pentru reprezentarea octală.

([Wakerly, 1990])

2.10 Se dă un întreg x în domeniul $-2^{n-1} \leq x \leq 2^{n-1} - 1$. Se definește numărul pozitiv $[x]$, ca fiind reprezentarea în complement față de doi a lui x: $[x] = x$ dacă $x \geq 0$ și $[x] = 2^n - |x|$ dacă $x < 0$, unde $|x|$ este valoarea absolută a lui x. Fie y un alt întreg în același domeniu ca și x. Să se demonstreze că regulile stabilite de adunare în complement față de doi sunt corecte prin demonstrarea faptului că următoarea relație este întotdeauna corectă:

$$[x + y] = ([x] + [y]) \operatorname{mod ulo} 2^n$$

([Wakerly, 1990])

2.11 Arătați cum se face adunarea numerelor reprezentate în cod BCD, formulând regulile de generare a transportului și de aplicare a factorului de corecție. Exemplificați pentru numere BCD formate din o singură cifră și din 2 cifre.

2.12 Arătați cum se face scăderea numerelor reprezentate în cod BCD, formulând regulile de generare a împrumutului și de aplicare a factorului de corecție. Exemplificați pentru numere BCD formate din o singură cifră.

2.13 Construiți câte o secvență de numărare în cod Gray pentru numere binare reprezentate pe 4 biți și respectiv pe 5 biți. Explicați care sunt avantajele utilizării codului Gray în aplicații.

2.14 Dacă $A = 0{,}101 \cdot 2^{101}$ și $B = 0{,}11 \cdot 2^{-11}$, să se calculeze expresiile $A + B$, $A - B$, și $A \cdot B$. Reprezentarea acestor expresii se face în virgulă mobilă.

([Friedman, 1986])

2.15 Reprezentați în virgulă mobilă numerele 123,321 și 345,543, pe care apoi le veți înmulți și împărți folosind reprezentările obținute.

([Ştefan, 2000])

2.16 Un cod BCD ponderat reprezintă fiecare cifră zecimală *N* printr-o secvență de 4 cifre binare $b_3 b_2 b_1 b_0$ cu ponderile w_3, w_2, w_1, w_0 astfel încât:

$$N = w_3 \cdot b_3 + w_2 \cdot b_2 + w_1 \cdot b_1 + w_0 \cdot b_0$$

a) Definiți un cod BCD cu ponderile 4,3,2,1 (codul definit de noi mai înainte este cel standard, având ponderile 8,4,2,1), astfel încât codul fiecărei cifre zecimale *N*, $0 \leq N \leq 9$, să fie complementul codului pentru cifra $9 - N$.

b) Câte seturi de ponderi cu această proprietate există, știind că nici una dintre ponderi nu poate fi negativă.

([Friedman, 1986])