UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiectarea Logică

25. Calculul binar

CALCULUL BINAR

In aceasta secțiune, vom vorbi despre aritmetica binara cu privire la operațiile de adunare, scădere și înmulțire a întregilor fără semn si a numerelor cu virgula fixa. Vom discuta întâi aritmetica zecimala.

Adunarea

La adunarea numerelor zecimale A si B, adunam cifrele corespunzătoare aceleiași locații, la care adunam eventualul transport de la locația precedenta. Exista posibilitatea ca suma sa fie de 2 cifre; acest lucru se întâmpla daca suma depășește 9 (baza-1). Suma poate fi scrisa ca CS, unde C este transportul. De exemplu, suma 7+5 este 12. Prin urmare, C este 1 iar S este 2. C si S satisfac relația:

$$S=(X+Y) \text{ MOD } b$$
$$C=(X+Y) \text{ DIV } b$$

unde X si Y sunt cele 2 cifre de adunat, iar b este baza. MOD returnează restul împărțirii lui X+Y la b, in timp ce DIV returnează catul.

La adunarea binara, cifrele folosite sunt 0 si 1. Rezultatul adunării a doua cifre binare este dat in Tabelul 1.7.1. El poate fi modificat pentru a include adunarea a 3 cifre binare. Tabelul 1.7.2 este modificat pentru a include si transportul. Adunarea a doua numere binare se face in pași intermediari, folosind tabelul de mai jos.

TABEL 1.7.1
Adunarea a doi biti binari, rezultatul necesita doi biti C si S

Cifrele originale	Suma	
A+B	***C***	***S***
0+0	0	0
0+1	0	1
1+0	0	1
1+1	1	0

TABEL 1.7.2
Adunarea a trei biti binari, rezultatul necesita doi biti C si S

Cifrele originale	Suma	
Transport+A+B	*C*	*S*
0+0+0	0	0
0+0+1	0	1
0+1+0	0	1
0+1+1	1	0
1+0+0	0	1
1+0+1	1	0
1+1+0	1	0
1+1+1	1	1

Exemplul 12:
Aduna următoarele numere binare: 11011 si 11001
Folosind Tabelul 1.7.2, avem rezultatul in Figura 1.7.1.
Discuția precedenta poate fi aplicata la adunare in baze arbitrare.
Următorul exemplu ilustrează acest lucru pentru doua numere in baza 5.

Exemplul 12a:

Aflați x+y, unde $x=(123.4)_5$ şi $y=(241.1)_5$
Exemplificam acest proces adunând cifrele individual si păstrând suma in zecimal; suma poate fi convertita in 2 cifre (in formatul CS) in baza 5.
Adunarea începe de la dreapta spre stânga. Pentru a determina suma, transportul trebuie de asemenea reținut. Suma este arătata in Tabelu 1.7.3. Prima coloana contine transportul si cifrele celor doua numere de la cea mai puțin semnificativa pana la cea mai semnificativa. Suma in zecimal este obținuta adunand cifrele de pe o linie. Transportul inițial este 0. Următoarele transporturi se obtin convertind suma zecimala in baza 5. De ex., prima linie produce suma 5, care e transformata in $(10)_5$. Transportul de 1 este trecut la linia următoare, iar procesul este repetat.

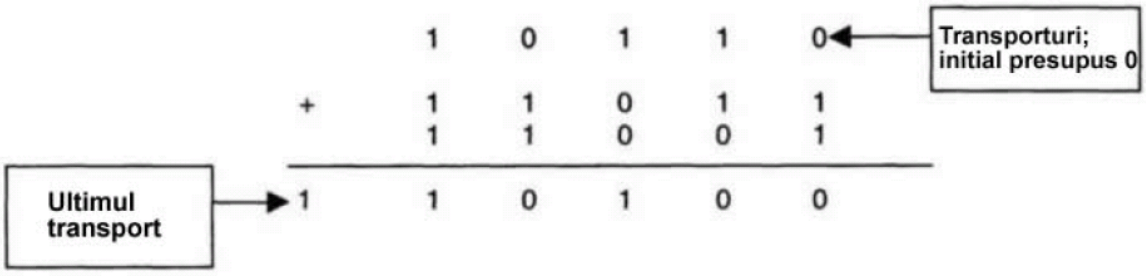


FIGURA 1.7.1
Adunarea a doua numere binare 11011+110001, linia de sus reprezinta transporturile

TABEL 1.7.3
Adunarea in baza 5, 123.4+241.1

Transport+x+y	Suma in baza zece	Suma in baza 5	
C	*S*		
0+4+1	5	1	0
1+3+1	5	1	0
1+2+4	7	1	2
1+1+2	4	0	4

TABEL 1.7.4
Adunarea in baza 16, 2397A.4+95CB.2

Transport+x+y	Suma in baza zece	Suma in baza 16	
C	*S*		
0+4+2	6	0	6
0+A+B	21	1	5
1+7+C	20	1	4
1+9+5	15	0	F
0+3+9	12	0	C
0+2+0	2	0	2

Suma numerelor se obtine din ultima coloana dupa ce includem punctul zecimal; suma este $(420.0)_5$.

Exemplu 14:
Suma x+y, unde x=$(2397A.4)_{16}$ si y=$(95CB.2)_{16}$ este data in Tabelul 1.7.4. Din Tabelul 1.7.4, $(2397A.4)_{16}$+$(95CB.2)_{16}$=$(2CF45.6)_{16}$. Sa se observe ca pentru y a fost adaugat un 0 in ultima linie.

Scaderea

Procesul de scădere într-o baza arbitrara deriva din scăderea numerelor zecimale.

Exemplul 15:
Afla $(1230015)_{10}$-$(1122124)_{10}$
Folosind scăderea obișnuita, rezulta notițele din Figura 1.7.2.
Aplicând cele mai de sus, obținem rezultatul din Figura 1.7.3. Așa cum poate fi văzut din figura, nu este nevoie de împrumuturi suplimentare.
Sa se retina ca la scăderea A-B, A se numește scazator iar B descazut.

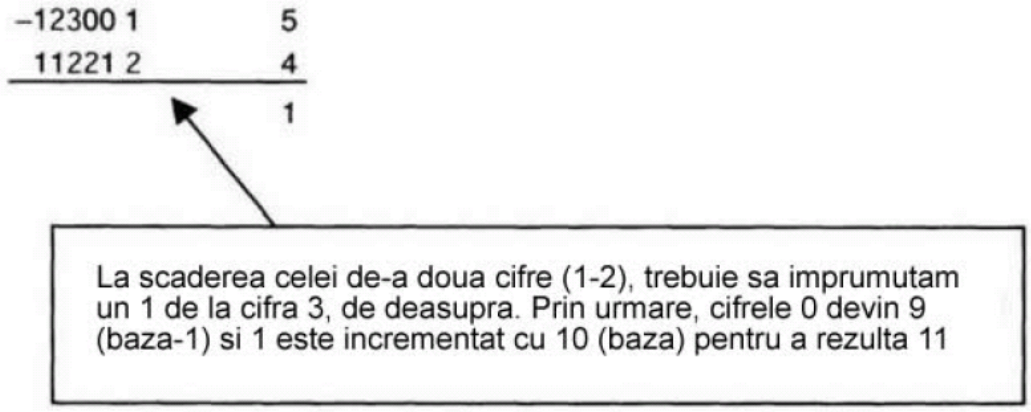


FIGURA 1.7.2

Scaderea in baza 10

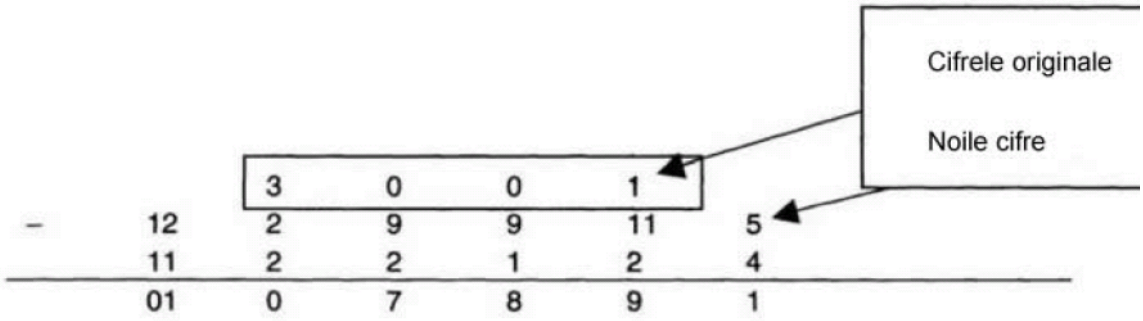


FIGURA 1.7.3

Scaderea in baza 10

Regula de mai sus poate fi generalizata pentru a scadea numere din baze diferite b. Pentru imprumut, cifrele 0 de la stanga sunt schimbate in b-1, cifra ce are nevoie de imprumut, a_i, e inlocuita cu (a_i+b), iar cifra ce a furnizat imprumutul, $a_{(i+j)}$, este decrementata cu 1.

Pentru demonstrare luam urmatorul exemplu.

Exemplul 16:

In acest exemplu gasim $(12034)_5$-$(11442)_5$.

Procedura de scadere este aratata in Figura 1.7.4 folosind pasii schitati mai sus.

Exemplul 17:

In acest exemplu calculam $(1100001)_2$-$(1011101)_2$.

Folosind algoritmul, obtinem Figura 1.7.5.

Exemplul 18:

Calculati $(1250A51)_{16}$-$(1170F31)_{16}$.

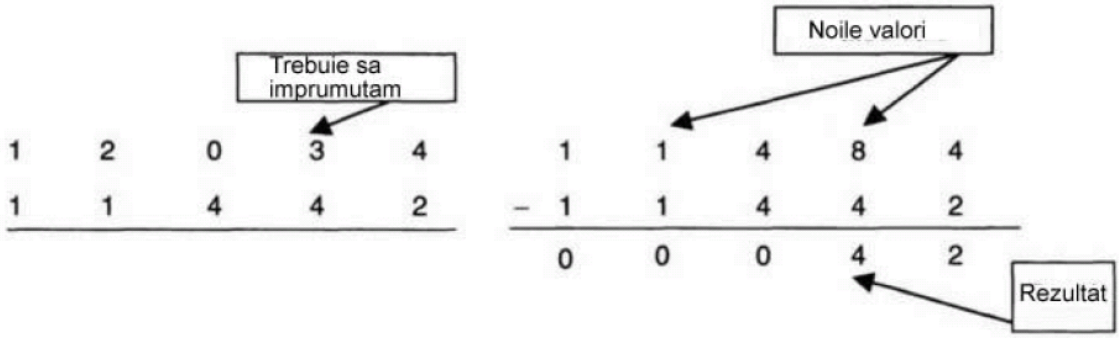


FIGURA 1.7.4

Scaderea in baza 5, 12034-11442

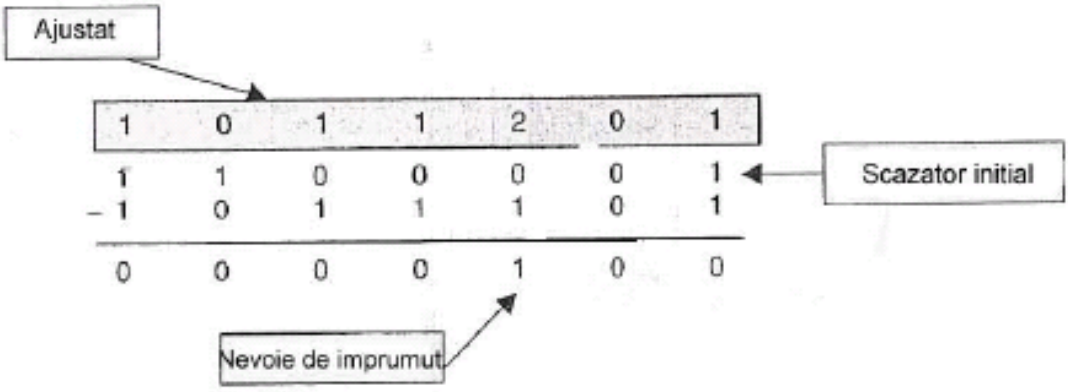


FIGURA 1.7.5

Scaderea in baza 2, 110001-1011101

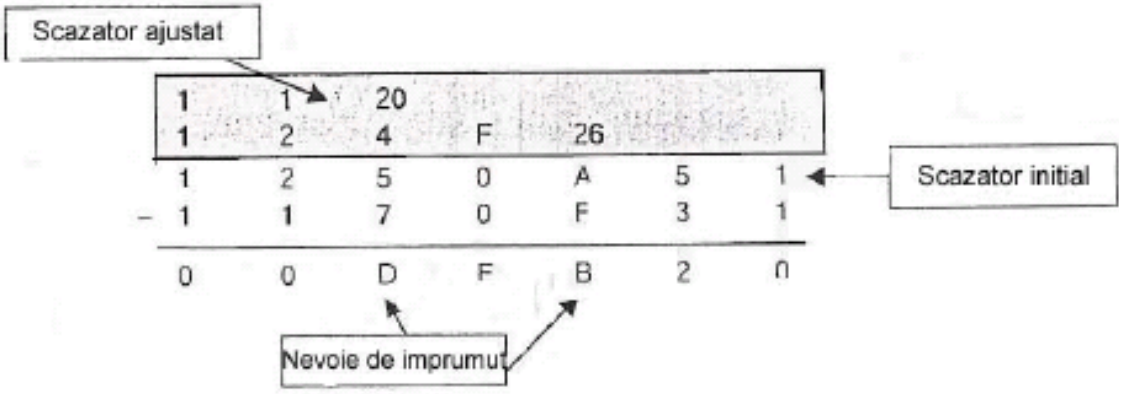


FIGURA 1.7.6

Scaderea in baza 16, 1250A51-1170F31

Scazatorii ajustati sunt aratati in zona umbrita din Figura 1.7.6. Prima linie este rezultatul primului imprumut; linia de deasupra ei este rezultatul celui de-al doilea imprumut.

Înmulţirea

Restricționam procesul de inmultire la numere binare, si ilustram procedeul printr-un exemplu

Exemplul 19:

Efectuand produsul $(11001)_2$ x $(1101)_2$.

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

Se observa ca deînmulțitul (11001) se copiază ca si cum bitul din înmulțitor (1101) este 1. Este înlocuit printr-un rând de zerouri daca bit-ul înmulțitorului este 0. Procesul este similar înmulțirii normale. La înmulțirea normala, formam produsele parțiale la fel cum am arătat mai sus. Fiecare produs parțial este mutat corespunzător cu un bit spre stânga. Pentru a obține produsul, adunam toate produsele parțiale după cum se observa in ultimul rând.

Complemente ale rădăcinii si ale rădăcinii diminuate

Discuția anterioara s-a ocupat cu reprezentarea numerelor nenegative. In partea următoare, vom discuta despre reprezentarea numerelor negative. Totuși, mai întâi, vom discuta doua tipuri de complemente care sunt folosite in aritmetica cu semn.

Reprezentarea in rădăcina diminuata a unui număr este determinata de trei parametrii: baza b, numărul de cifre al numărului n, si numărul propriu-zis N. Pentru un număr N, reprezentarea in rădăcina diminuata este

$$(b^n - 1) - N$$

Reprezentarea se numește complementul lui (b – 1). De exemplu, in baza 10, reprezentarea se numește complementul lui 9. In mod similar, in baza 2 reprezentarea in rădăcina diminuata se numește complementul lui 1.

Exempul 20:

Formati complementul față de 9 a numărului zecimal 123.

Rezolvare :

Din moment ce N=123, numarul de cifre, n, este 3 si

$$b^n - 1 = 10^3 - 1 = 999$$

Din acest motiv

$$(b^n - 1) - N = 999 - 123 = 876$$

Ce este $(b^n - 1)$? Din exemplul de mai sus, observam ca $(b^n - 1)$ este un număr cu *n* cifre, fiecare cifra fiind egala cu *b-1*, in consecința, pentru a obține complementul rădăcinii diminuate a unui număr N, extragem fiecare cifra N din (b - 1). Rezultatul obținut este in rădăcina diminuata.

Exemplul 1.8.2

In acest exemplu, formam complementul lui 1 a numărului binar 10110. Folosind observația anterioara, avem

$$11111 - 10110 = 01001$$

ca fiind complementul 1 a lui 10110.

Observam ca complementul rădăcina diminuata a unui număr se obțină din fiecare cifra luata individual. Suma cifre si cifra corespunzătoare in complementul rădăcina diminuata trebuie sa se adune la (b – 1). In cazul complementul lui 1 , obținem complementul complementând fiecare cifra a numărului. Pentru fiecare cifra din număr, suma biților si bitul corespunzător in complementul lui 1 trebuie sa ne duca la rezultatul (b – 1 = 1). Aplicam regula unei alte baze, folosind exemplul de mai jos.

Exemplul 1.8.3

Găsiți complementul lui 4 al numărului $(1234)_5$.

Folosind observația de mai sus, complementul lui 4 este 3210 deoarece :

1 + 3 = 4 , 2 + 2 = 4 , 1 + 3 = 4 si 0 + 4 = 4

In mod asemănător, reprezentarea complementului rădăcinii este determinata de trei parametrii : baza *b*, numărul de cifre al numărului *n*, si numărul propriu-zis *N*. Reprezentarea se numește : *reprezentarea complementului lui b*. Se numește complementul lui 10 in baza 10 si complementul lui 2 in baza 2. Pentru un număr N, reprezentarea complementului rădăcinii a numărului este :

$$\text{complementul lui b} = \begin{cases} \text{Caz 1. } b^n - N \text{ , daca } N \neq 0 \\ \text{Caz 2. } 0 \text{ , daca } N = 0 \end{cases}$$

Observam ca : $b^n - N = ((b^n - 1) - N) + 1$, in consecinta, pentru un numar dat N, avem complementul lui *b* al complementului lui $N = (b - 1)$ al lui $N + 1$.

Exemplul 1.8.4

Formați complementul lui 10 si complementul lui 2 al numerelor 0821_{10} respectiv 001011_2 .

Pentru 0821_{10} , n = 4; complementul lui 10 este : $10^4 - 0821 = 1 + 9999 - 0821 = 9179$. Pentru 001011_2 , n=6; complementul lui 2 este $2^6 - 001011 = 1 + 111111 - 001011 = 110101$.

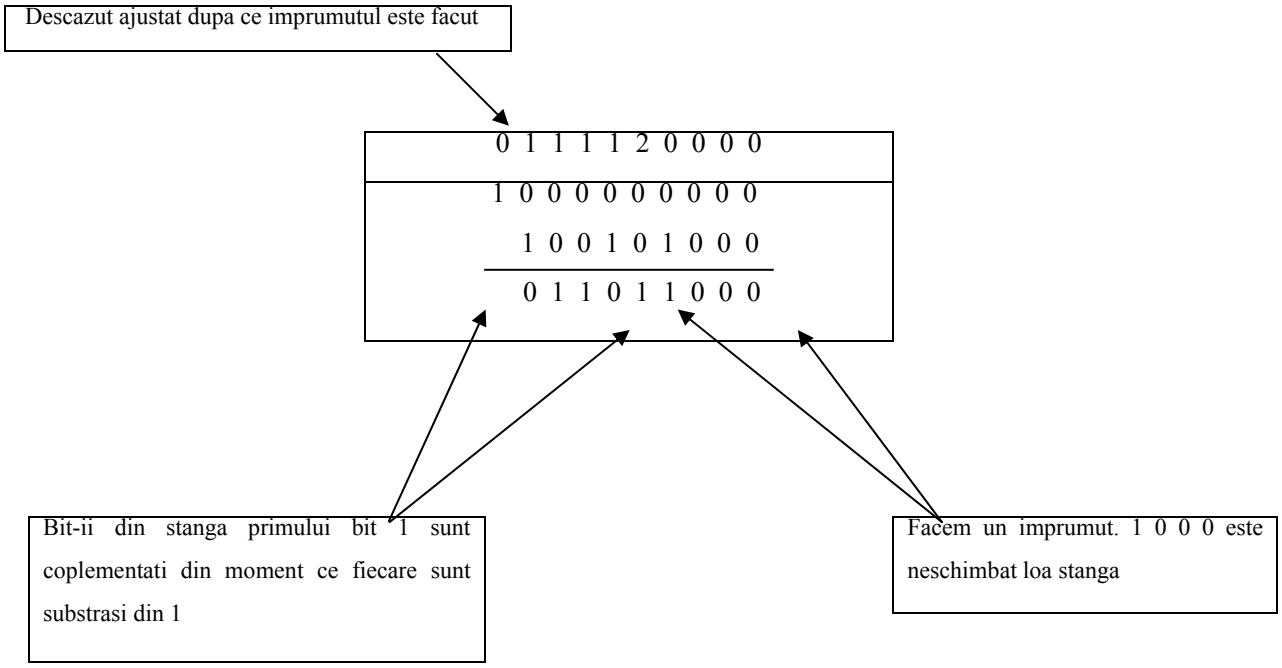


Figura 1.8.1

Formarea complementţilor lui 2. Plecând de la definiție, nu are loc nici o schimbare a celor mai multe zerouri consecutive din partea stânga in număr si a primului bit 1 care începe împrumutul. Toți biţii rămași sunt complementați.

O metoda alternative de găsire a complementului lui 2 se folosește de sustragerea despre care am discutat anterior. Fiind dat un număr, N, cu m zerouri consecutive urmate de un 1 in poziția m + 1. După cum arata si figura 1.8.1 cu N = 100101000 drept exemplu, fiecare din primii m biți rămâne neschimbat (0 – 0 = 0) atunci când formam complementul 2 al lui N. Al (m + 1)-lea bit este extras din 2, din moment ce acest bit inţiază un împrumut, si ca rezultat, rămâne neschimbat (2 – 1 = 1). Biţii ramași sunt extrași din 1, in consecința, fiecare bit este complementat. Exemplul de mai jos arata acest lucru.

Exemplul 1.8.5

Formati complementul 2 al numarului 100101000_2 .

Aratam detaliile procedurii effectuate mai sus in figura 1.8.1. Complementul lui 2 este aratat ca fiind ultimul rand al numerelor binare.

Reprezentarea numerelor negative

In capitolul 1.5.2, am discutat despre overflow (depășire superioara) si underflow (depășire inferioara) ca fiind condiții care rezulta din stocarea unui număr care este fie prea mare sau prea mic pentru a începăea in registru predefinit. Folosind numere întregi fara semn, consideram diferența $0011_2 - 0110_2$. Rezultatul este -3_{10} , care este prea mic pentru a incapea intr-un registru de 4 biti. In acest capitol, vom discuta prezentarea numerelor negative. Din moment ce calculatoarele folosesc numai numere binare, numerele negative sunt reprezentate folosind numai cifre binare.

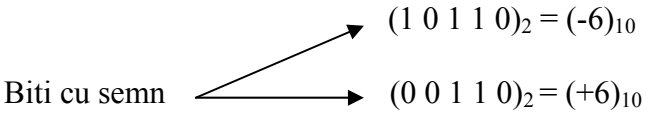


Figura 1.9.1
Reprezentarea prin semn şi mărime ,cel mai significant bit folosit pentru a reprezenta semnul unui numar.

Cele trei reprezentări

Consideram cele trei reprezentari :

1. Magnitudine cu semn
2. Complementul radacinii diminuat
3. Complementul radacinii

In reprezentarea magnitudine cu semn, numărul este format dintr-o parte cu semn si o parte magnitudine. Acest lucru este asemănător modului in care reprezentam numerele negative. De exemplu "-123" reprezintă un numar negativ in baza 10. Semnul este "-", iar magnitudinea este "123".

Cand reprezentam numarul in binar, din moment ce bit-ii pe care ii putem folosi sunt 0 si 1, numin unul din biti sa reprezinte semnul. Bit-ul este MSB-ul cu 0 reprezentand semnul numerelor nenegative si 1 semnul numerelor negative. Din acest motiv numerele -6 si +6 se reprezinta ca in figura 1.9.1.

Numerele negative pot fi deasemenea reprezentate folosind reprezentarea complementara prezentata in capitolul anterior. (Avantajul reprezentarii va fi prezentat in Capitolul 5 cand vom discuta despre design-ul unitarilor aritmetice). In ambele reprezentari, MSB-ul este folosit ca un bit de semn cu 1 indicand numere negative si 0 indicand numere pozitive. Cand alte baze sunt folosite pentru reprezentarea numerelor negative folosim 0 si (b – 1), pentru numere pozitive respectiv negative. Pentru a gasi reprezentarea unui numar negativ, formam radacina corespunzatoare, sau complementul radacinii diminuat.

Exemplul 1.9.1

Presupunând numere de patru cifre, găsiţi complementul 10 si complementul 2 al reprezentării numerelor $(-23)_{10}$ respectiv $(-11)_2$.

Rezolvare :

Conform definitiei, avem complementul lui 10 al lui $23 = 10^4 – 23 = 9999 – 23 + 1 = 9977$, in consecinţă, -23 in aceasta reprezentare este 9977. Observam ca cifra 9 arata ca numărul este negativ. Pentru complementul lui 2, folosind observaţia anterioara pentru 00112 obtinem 1101 ca fiind reprezentarea complementului lui 2 al lui 0011_2 (11_{10}) , in consecinta, (-11_{10}) in aceasta reprezentare este 1101.

Exemplul 1.9.2

Scrieti reprezentarea complementului lui 10 a numerelor in baza 10, de doua cifre, pozitive si negative.

Tabelul 1.9.1

Numere negative de doua cifre si valoarea corespunzatoare in baza 10

Valoarea N + complementul lui b al lui N = 100	Valoarea zecimala	N + complementul lui b al lui N =100	Valoarea zecimala
? + 90	-10	? + 95	-5
? + 91	-9	? + 96	-6
? + 92	-8	? + 97	-7
? + 93	-7	? + 98	-8
? + 92	-6	? + 99	-9

Observatie : In acest tabel 90 reprezinta numarul $(-10)_{10}$. Analog celalalte.

Rezolvare :

Din moment ce cea mai semnificativa cifra este folosita pentru semn, avem in total 10 numere nenegative (00, 01, 02, … , 09). Pentru reprezentarea negativa cea mai semnificativa cifra este 9. Reprezentarile negative sunt 90, 91, 92 , … 99. Pentru a gasi numarul negativ asociat fiecarei reprezentari, folosim urmatoarea ecuatie:

$$N + \text{complementul lui b a lui } N = b^n$$

Unde $b^n = 100$. Tabelul 1.9.1 foloseste ecutia de mai sus.

Observam ca reprezentarea negativa a lui -10 este 90, si ca cel mai mare numar pozitiv este +9, cu reprezentarea 09 (0 indicand ca numarul este pozitiv si 9 fiind magnitudinea lui). Aceasta este o proprietate a complementului lui b unde cel mai mic numar negativ nu are corespondent in representarea pozitiva.

1.9.2 Intervalele numerelor

Pentru reprezentarea magnitudinii cu semn si a unui intreg de n biti, cu semn, numarul total permis de combinatii binare este 2^n . Din moment ce unul dintre biti este folosit ca bit de semn, intervalul numerelor pozitive este (+0, $2^{(n-1)}$ -1). In mod analog, intervalul pentru numere negative este (-($2^{(n-1)}$ -1), -0) . Observam ca sunt doua reprezentari diferite ale lui 0(una pozitiva si una negativa). Mai observam ca rezultatul se aplica pentru numere in orice baza cu intervalul numerelor positive (+0, $b^{(n-1)}$ -1) si intervalul numerelor negative (-($b^{(n-1)}$ -1), -0).

Pentru reprezentarea complementului lui b, generalizam rezultatul din subcapitolul precedent. Pentru asemenea numere, intervalul numerelor nenegative este (+0, $b^{(n-1)}$ -1) unde n este numarul total de cifre si b este baza in care este scris numarul. Intervalul negativ este (-($b^{(n-1)}$ -1), -0).

Tabelul 1.9.2
Cele trei reprezentari

Reprezentare	Forma	Interval pozitiv	Interval negativ	Observatii	MSD
Magnitudine cu semn	Sign(magnitude)	(+0,$b^{(n-1)}$ -1)	(-($b^{(n-1)}$-1),-0)	2 zeoruri	0,(b-1)
Radacina diminuata	($b^{(n-1)}$ -1) - N	(+0,$b^{(n-1)}$ -1)	(-($b^{(n-1)}$-1),-0)	2 zerouri	0,(b-1)
Radacina	b^n - N	(+0,$b^{(n-1)}$ -1)	(-($b^{(n-1)}$-1),-0)	1 zerou	0,(b-1)

Tabelul 1.9.3
Cele trei reprezentari asuprea unui operand scris pe 3 biti

Valoarea zecimala	Magnitudine cu semn	Complementul lui 1	Complementul lui 2
+0	000	000	000
+1	001	001	001
+2	010	010	010
+3	011	011	011
-0	100	111	Nu exista
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	Nu exista	Nu exista	100

Observam ca sunt doua reprezentari pentru 0 in semn si in reprezentarea complementului lui 1 . Mai observam ca reprezentarea complementului lui 2 contine un numar negative , (- 4) care nu are corespondent in randul numerelor positive.

Reprezentarea complementelor este una ciudata din puntul de vedere al utilizatorului. Totusi ele sunt potrivite pentru folosirea in procesele calculatorului,

dupa cum vom discuta in capitolul 5. In zilele noastre, calculatoarele reprezinta intregi cu semn in complementul lui 2 unde numerele negative sunt stocate astfel.

1.10 Codificarea si codurile binare

In tabelul 1.9.3 am reprezentat intregi pozitivi si negativi in trei forme diferite. Fiecare reprezentare este numita, in general, un cod.

Un cod este atribuire care asociaza cu fiecare element dintr-un set de obiecte un alt element, numit *code word*(cuvand de cod), intr-un alt set de obiecte. De exemplu, in reprezentarea complementului lui 2 ne putem gandi la atribuirea 100 ca fiind cuvantul de cod al intregului , in baza zecimala, -4. In mod asemanator, 101 este cuvantul de cod pentru intregul, in baza zecimala, -3. Setul de obiecte considerat in tabel sunt intregii in baza zecimala in intervalul (-4, +3). Codul complementului lui 2 este dupa cum se vede. Procesul de determinare al codului asociat cu un obiect dat se numeste codificare. De la un cuvant de cod dat, procesul de determinare a obiectului original de numeste decodificare.

Tabelul 1.10.1

BCD si Exess 3 pentru cifrele in baza 10

Cifre in baza 10	Codul BCD	Codul Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

In aceasta parte vom prezenta 4 tipuri de coduri: codul binar zecimal (BCD), codul Excess-m, codul Gray si codul caracterelor. Pentru notatie se va folosi *X(n)* pentru a reprezenta codul obiectului *n*, când codificarea *X* este aplicata. De exemplu, BCD(5) reprezintă codificarea BCD pentru cifra 5.

1.10.1 Codul BCD

In aceasta schema de codare, se asociază fiecărei cifre zecimale un cod binar. Cum numărul de cifre este 10 (de la 0 la 9), numărul minim de cifre binare (biti) necesar intr-un cod este 4. Codul BCD provine de la reprezentarea polinomiala a numerelor binare. Codificarea numerelor zecimale 0,1 si 9 este data de secventele 0000, 0001 si respectiv 1001. Coloanele 1 si 2 din tabelul 1.10.1 prezinta cele zece cifre zecimale si corespondenta in cod BCD. Codificarea BCD necesita un numar mai mare de biti decat precedentele scheme de codificare prezentate.

Codul este numit cod ponderat deoarece cifra asociata unei secvente de cod date este suma ponderata a cifrelor ce formeaza secventa de cod. Ponderile sunt 8, 4, 2 si 1, corespunzatoare MSB, respective LSB. Codul este numit si 8421.

Exemplu 1.10.1

Gasiti codul, BCD(127), pentru 127_{10} .

In reprezentarea BCD, codificarea se obtine prin asocierea la fiecare cifra a corespondentului in cod BCD. Codurile BCD ale fiecarei cifre sunt apoi concatenate. In consecinta, avem:

127 —codificarea fiecarei cifre→ 0001 0010 0111 —concatenare→ 000100100111

1 2 7

Astfel, BCD(127) = 000100100111

Exemplu 1.10.2

Decodificati secventa BCD 0000100100110001 si gasiti numarul original.

Pentru a gasi numarul echivalent codului de mai sus se grupeaza bitii in grupuri de cate 4. Se inlocuieste fiecare grupare cu cifra corespunzatoare. Cifrele obtinute sunt in final concatenate pentru a obtine rezultatul, asa cum este prezentat mai jos.

0000100100110001 —grupare→ 0000 1001 0011 000 —concatenare→ 0931

Asa cum se poate vedea din exemplul de mai sus, transformarea numerelor zecimale in cod BCD este mai simpla decat aplicarea algoritmilor precedenti de gasire a echivalentului in cod binar. Informatia intr-un computer poate fi introdusa prin utilizarea tastaturii. Ca urmare, urmatoarele procese au loc:

1. Informatia alfanumerica (cifre si caractere) este codificata pentru a forma un sir binar.
2. Computerul proceseaza informatia codificata
3. Computerul decodifica rezultatul in alfanumeric astfel incat sa poata fi afisat pe un dispozitiv de iesire, cum ar fi monitorul. Aceste operatii sunt prezentate in figurile 1.2.1 si 1.2.2

Pentru operațiile de intrare/iesire intense, utilizarea unui cod similar cu BCD va accelera procesul pana cand majoritatea calculelor vor fi efectuate in pasii 1 si 3 prezentați mai sus. Ca urmare, unele computere pot avea componente hardware pentru efectuarea transformărilor BCD. Aceste componente sunt mai lente decat dispozitivele hardware de procesare a informației bazata pe codul binar direct prezentat mai devreme. Totuși, accelerarea este datorata frecventei de executie a pașilor 1 si 3.

1.10.2 Codul Excess-m

Codul Excess-m se obtine dintr-o secventa de cod, *c*, adunand numarul *m* la cod. De exemplu, codul Excess-3 pentru BCD este obtinut adunand 3 la fiecare secventa de cod. Coloanele 1 si 3 din tabelul 1.10.1 prezinta cifrele zecimale si echivalentul lor in cod Excess-3 pentru reprezentarea BCD.

Codul Excess-3 nu este un cod ponderat si este un exemplu de cod auto-complemenatr. Un cod este auto-complementar daca pentru orice cifra de la 0 la 9 se poate obtine complementul cifrei la 9 prin negarea individuala a codului. De exemplu, in tabelul 1.10.1, codul Ecess-3 pentru cifra 3 este
:

BCD(3)+3 = 0011 + 0011 = 0110

Reprezentarea in Excess-3 a complementului la 9 (6) se obtine prin negarea fiecarui bit din 0110, obtinandu-se 1001. O cercetare a tabelului confirma acest fapt.

1.10.3 Codul Gray

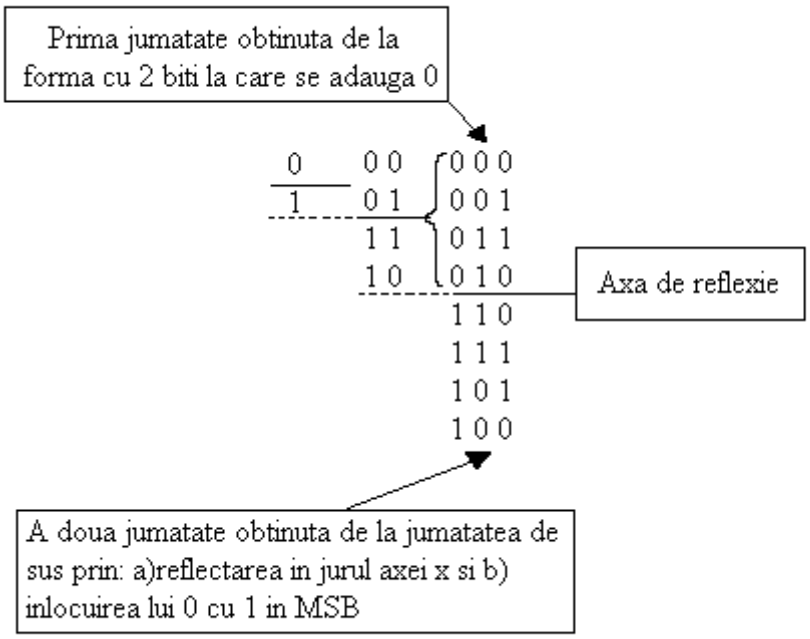
Urmatorul cod prezentat este codul Gray. Este un tip special de cod, numit ciclic. Un cod este ciclic daca prin aplicarea unei transformari circulare a fiecarei secventa de cod se obtine o alta secventa. Are proprietatea ca distanta dintre doua secvente de cod consecutive este egala cu 1. Distanta dintre oricare doua secvente este egala cu numarul de biti prin care acestea difera.

Codul Gray este un membru al clasei de coduri numite coduri reflectoare. Codurile cu n+1 biti este obtinut din codurile cu n biti, dupa cum urmeaza. Numarul de coduri de bit n+1 este de doua ori mai mare decat decat pentru n biti. Prima jumatate a codului n+1 se obtine din codul n prin adaugarea unui 0 la stanga codului MSB (figura 1.10.1) Celelalte coduri se obtin prin reflectarea primei jumatati in jurul axei, asa cum se aratas in figura 1.10.1 si prin inlocuirea in MSB a lui 0 cu 1.

Codul Gray este utilizat in cazul in care este important sa nu se obtina rezultate intermediare incorecte. Sa consideram doua coduri consecutive BCD pentru 7 si 8. Pentru a trece din 0111 in 1000 sunt necesare doua schimbari in pozitia bitilor. Daca aceste schimbari nu se produc simultan, atunci unele rezulate intermediare(nedorite) se pot petrece; de exemplu schimbarea primului bit inaintea celorlalti genereaza codul intermediar 0110. Vom vedea mai tarziu cum aceste schimbari intermediare pot cauza probleme in proiectarea digitala.

Inainte de a parasi aceasta sectiune, vom discuta o schema finala de codificare a caracterelor.

`



1.10.4 Coduri pentru caractere

In figura 1.2.1, computerul primeste date codificate pentru cifre si litere, precum si semen de punctuatie si alte caractere de control. Pana acum, discutia despre codificare s-a limitata la numere. In continuare vom avea in vedere codificare binara alfanumerica. Aceasta include coduri pentru cifre, litere si caractere speciale, precum si caractere de control. Exemple de caractere speciale sunt ^, + si $. Caractere de control sunt tastele “space” si “delete” de la tastatura.

Tabelul 1.10.2 prezinta o schema de codificare alfanumerica, American Standard Code for Information Interchange (ASCII, pronuntat “ask-ee”). O alta codificare, mai putin comuna, este Extended Binary Coded Decimal Interchange Code (EBDIDIC).

Tabel 1.10.2
Caractere ASCII

(linii in hexa) $b_3b_2b_1b_0$	$b_6b_3b_4$ (coloane in octo) 0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w

	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	RS	,	<	L	\	l	|
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	?	N	^	n	~
	F	SI	US	/	?	O	_	o	
DEL									

Nota: abrevierile sunt NUL-null; SOH-start of heading; STX-start of text; ETX-end of text; EOT-end of transmission; ENQ-enquiry; ACk-acknowledge; BEL-bell; BS-backspace; HT-horizontal tab; LF-line feed; VT-vertical tab; FF-form feed; CR-carriage return; SO-shift out; SI-shift in; SP-space; DLE-data link escape; DC1-device control 1; DC2-device control 2; DC3-device control 3; DC4-device control 4; NAK-negative acknowledge; SYN-synchronize; ETB-end of transimission block; CAN-cancel; EM-end medium; SUB-substitute; Esc-escape; FS-file separator; GS-group separator; RS-record separator; US-unit separator; SP-space; Del-delete or rub out.

1.11 Numere in virgula mobila

Numerele reale sunt reprezentate in computer fie in virgula fixa, fie in virgula mobila. Formatul de virgula mobila este folosit pentru a reprezenta numere foarte mari sau foarte mici, necesare in aplicatii stiintifice. De exemplu, numarul 7×2^{64} reprezentat in binar este $(111)_2$ urmat de 64 de biti 0. Ca alternativa, reprezentarea in virgula mobila este mult mai eficienta. Se stocheaza reprezentarea binara a exponentului (64) si a partii intregi (7); Baza este implicita. Rezultatul este

1 1 1	1 0 0 0 0 0 0
Partea intreaga	Partea exponentiala

In general, un numar in virgula mobila, F, este reprezentat ca:

$$F=+-M \times rE$$

Unde M, mantisa sau signifiantul , este un numar fix; E, exponentul (numit si caracteristica) este intreg, iar r este baza

In general, un numar fix poate fi scris in virgula mobila. Aceasta reprezentare nu este unica.

Exemplul 1.11.1

Reprezentați numărul fix 1.23 in virgula mobila, folosind exponenții 0, 1 si 2

$$\begin{aligned} 1.23 &= 1.23 \times 10^0 \\ &= 0.123 \times 10^1 \\ &= 0.0123 \times 10^2 \end{aligned}$$

In continuare vom analiza reprezentarea binara a numerelor in virgula mobila

1.11.1 Reprezentarea binara a numerelor in virgula mobila

In binar, reprezentarea in virgula mobila a numerelor este

$$F=(-1)^s M \times 2^E$$

Unde M si E au aceeași semnificație ca mai sus, dar M reprezentând numai partea fractionara; s este o reprezentare pe un singur bit a semnului cu s=0 pentru numere pozitive si s=1 pentru numere negative. Domeniul numerelor in virgula mobila depinde de numărul de biti alocat pentru fiecare parte a mantisei si exponentului

Exemplu 1.11.2

Fiind dat un cod pe 7 biti, 1110101, efectuati decodificarea presupunand ca reprezinta:

1 Un intreg fara semn
2 Un intreg cu semn in complement la 2
3 Un caracter in codul ASCII
4 Un numar in virgula mobila cu M pe 3 biti

Solutie:

1 Ca un intrg fara semn, codul reprezinta $2^6 + 2^5 + 2^4 + 2^2 + 2^0 = (117)_{10}$

2 Ca un numar cu semn in complement la 2, codul reprezinta $-(128-117)=(-11)_{10}$

3 Ca un caracter ASCII din tabelul 1.10.2, codul reprezinta litera “u”

4 In final, ca un numar in virgula mobila avem:

1	1 1 0	1 0 1
Semn	Exponent	Mantisa

Sau, codul reprezinta

$$\begin{aligned} (-1)^s M \times 2^E &= -(0.101)_2 \times 2^{(110)}2 \\ &= -0.625 \times 2^6 \end{aligned}$$

In exemplul precedent, am vazut ca acelasi cod poate fi folosit pentru a reprezenta diferite date. Numărul maxim de obiecte ce poate fi codificat nu depaseste rangul

maxim impus de dimensiunea codului (2^7). Drept urmare, daca sunt reprezentati întregi fara semn, zona de interes este 0 – 127. Similar, pentru numere cu semn in complement la 2, limitele sunt -64 la 63. In final, in reprezentarea in virgula mobila, cel mai mare număr corespunde secvenței 0111111, care este 0.875 x 128 = 112. Similar, cel mai mic număr reprezentabil este -112.

Cum reprezentarea in virgula mobila este folosita pentru a reprezenta numerele reale si cum numărul maxim de obiecte ce pot fi codate este 128, rezulta ca avem multe numere reale intre -112 si +112 cărora nu le corespund o reprezentare in virgula mobila. Ca atare, crescând mărimea intervalului scade numărul de reprezentari posibile intr-un sub-interval dat.

Exemplu 1.11.3
Din exemplu anterior, găsiți intervalul de numere in virgula mobila cand reprezentarea exponentului creste cu 1 bit, iar campul mantisei scade cu 1. De asemenea pentru câmpul exponentului crescut cu 2 biti si al mantisei scazut cu 2.

Soluție: Cu noile formate, intervalul maxim pozitiv pentru schimbarea de 1 bit este

$$\begin{aligned}(-1)^s M \times 2^E &= (0.11)_2 \times 2^{(1111)_2} \\ &= 0.75 \times 2^{15} \\ &= 24576\end{aligned}$$

Intervalul maxim pentru schimbarea de 2 biti este

$$\begin{aligned}(-1)^s M \times 2^E &= (0.1)_2 \times 2^{(1111)_2} \\ &= 0.5 \times 2^{31} \\ &= 1073741824\end{aligned}$$

Cel mai mic interval negativ este obținut folosind aceeași reprezentare cu bitul de semn, presupunând o valoare de 1. Putem reprezenta numai un număr maxim de 128 de numere reale in intervalul de mai sus.

1.11.2 Reprezentare in virgula mobila normalizata si polarizata

Reprezentarea precedenta poate conduce la diferite cuvinte de cod care reprezintă acesali obiect codificat. Ca exemplu, cele doua reprezentari in virgula mobila, $0.010 \times 2^{(101)_2}$ si $0.001 \times 2^{(110)_2}$ reprezinta acelasi numar zecimal, $(8)_{10}$. Pentru a inlatura aceasta ambiguitate, numerele in virgule modbila sunt reprezentate in forma normalizata. Un numar in virgula mobila este normalizat daca cel mai semnificant bit al mantisei este 0.

In plus fata de reprezentarea unui numar in forma normala, exponentul unui numar in virgula mobila este reprezentat in general in forma unui intreg polarizat cu semn (intreg excess-m). Forma polarizata este $2^{(n-1)}$-1, unde n este numarul de biti in campul exponentului. Vom ilustra cele scrise mai sus in doua exemple.

Exemplul 1.11.4
Codificati numarul in virgula fixa $(110.10)_2$ intr-o reprezentare virgula mobila polarizata si normalizata cu campurile corespunzatoare exponentului si mantisei egale cu 4 biti si respectiv 6 biti.

Solutie: In primul rand vom scrie numarul 110.10 in virgula mobila normala

$$(110.10)_2=(0.110100)_2 \times 2^3$$

Apoi adaugam 7 la exponent pentru a obtine un exponent polarizat $(10)_{10}=(1010)_2$
Atunci reprezentarea devine

0	1010	110100
S	E	M

Exemplul 1.11.5
Dat fiind numarul in virgula mobila normalizat si polarizat de mai jos, aflati valoarea sa in baza 10.

0	110010	1010
S	E	M

Solutie: Bitul de semn indica faptul ca numarul e pozitiv. Mantisa sa este 0.1010, adica

$$1/2+1/8=0,625$$

Exponentul sau este obtinut prin decrementarea sa cu 2^{6-1}-1=31. Astfel obtinem 010011 care este $(19)_{10}$. Ca urmare, valoarea in baza 10 a numarului este 0.625×10^{19}.