UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2007-2013

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

Proiectarea Logică

1. Descrierea semnalelor

DESCRIEREA SEMNALELOR

În calculatoare orice entitate logică este reprezentată printr-un semnal. Există, în general semnale analogice şi semnale digitale. Semnalele analogice pot lua o infinitate de valori dintr-un interval precizat. Semnalele digitale pot lua doar doua valori dintr-un interval de tensiuni.
Dacă se consideră intervalul de tensiuni cuprins între 0V şi +5V, semnalele digitale vor lua doar două valori. Prima situată undeva în apropierea valorii maxime, +5V, iar cealaltă în apropierea valorii 0V. Un semnal analogic din acelaşi interval de tensiuni poate avea orice valoare din interval
Termenul *digital* caracterizează, în genere, datele dintr-o mulţime. Se spune ca o mulţime de date este digitala daca ea conține un număr finit de elemente. Exemple de date digitale sunt mulțimile {0,1}, {pornit,oprit}, {rosu, albastru, verde} si {x: x este cifra zecimala}. In fiecare din aceste exemple, mulțimea de date este digitala – din moment ce numărul de elemente conținute este finit.
Mulțimea de date este analogă daca este formata dintr-un interval continuu de elemente. Astfel, mulțimea de date conține un număr infinit de elemente, având in vedere faptul ca intervalele continue cuprind o infinitate de elemente. Exemple de date analog sunt multimea numerelor reale, multimea tuturor culorilor si multimea numerelor reale intre 0 si 10.
De multe ori se aproximează datele in format analog prin conversia in digital. De exemplu, timpul este analog. Cand se comunică ora exacta, se spune ora si minutul. Un exemplu este afisajul orei pe ecranul calculatorului.

Codarea

Se observa ca multimea datelor introduse în calculator de la tastatura este digitala deoarece este compusa dintr-un numar finit de elemente.
Calculatorul proceseaza date in format binar (compuse din 0 si 1), astfel ca trebuie urmati cativa pasi:

1. Datele introduse la tastatura sunt codate in secvente de biti unde literelor, cifrelor si altor simboluri li se asociaza un cod binar – cuvant cod. Codarea este o mapare care ataseaza fiecarui obiect dintr-o multime un unic element (cuvant cod) dintr-o alta multime. In acest caz, multimea de obiecte este multimea de date introduse de la tastatura. Vom vrea sa asociem o secventa de cod binar tuturor datelor introduse de la tastatura
2. Al doilea pas in acest proces este executarea de catre calculator a sarcinii dorite. Aceasta are loc in unitatea centrale de procesare (UCP), care include circuite electrice speciale ce realizeaza sarcinile.

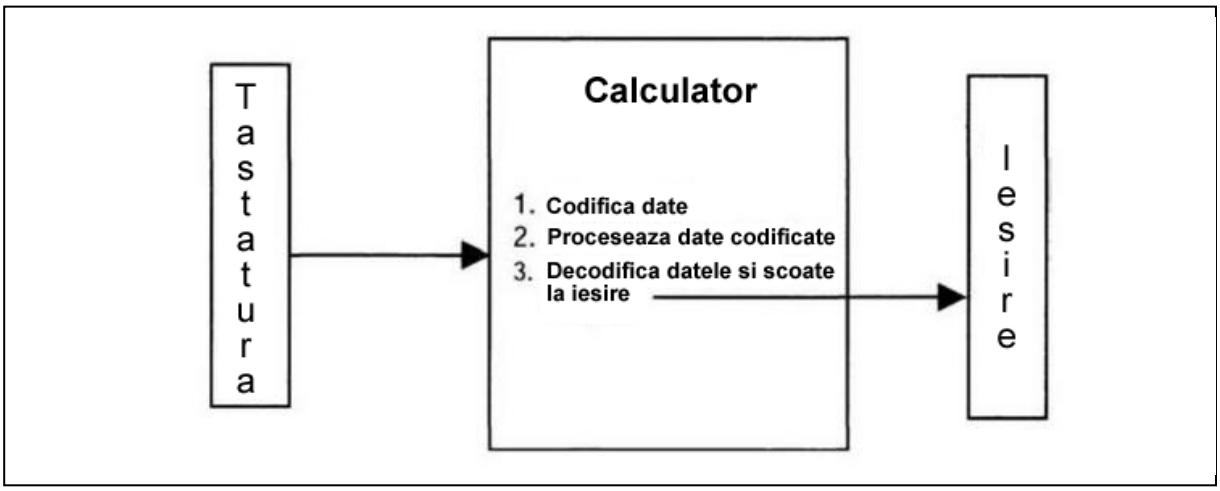

Figura 1. Cei trei pași – codare, procesare si decodare

3. Ultimul pas este conversia rezulatului binar inapoi in format inteligibil de catre utilizator, de exemplu in cifre digitale si litere din alfabet. Aceasta etapa este numita decodare si este inversul codarii. Figura 1 ilustrează acești 3 pasi.

In continuare vom discuta detaliile celor trei pasi. Din moment ce calculatoarele proceseaza date binare, acesta va fi punctul de pornire.

Sistemul Numeric Poziţional

In sistemele numerice pozitionale, un numar *N* este caracterizat printr-o baza, *b*, numita si radacina, si coeficienti (0, 1, ..., (b-1)), care alcatuiesc numarul. In multimea numerelor zecimale (baza 10), baza este implicita si poate fi omisa. Cand sunt luate in calcul alte baze, acestea trebuie specificate explicit in reprezentarea ca $(N)_b$, sau N_b, unde b este baza.
De exemplu, in baza 10 coeficientii lui *N* sunt 0, 1, ...,9. Similar, in baza 5 coeficientii unui numar sunt 0, 1, 2, 3 si 4, iar pentru baza 16 avem cifrele zecimale de la 0 la 9, la care se adauga literele alfabetului A, B, C, D, E si F, reprezentant numerele zecimale 10,11, 12, 13, 14 respectiv 15. Simbolurile sunt folosite pentru ca baza numarului depaseste baza 10. Primele 6 litere ale alfabetului sunt folosite de in completarea celor 10 cifre digitale. Consideram doua formate, numerele cu punct radacina baza si numerele fara punct radacina.

Numere întregi

Pentru un numar dat $(N)_b = n_i\ n_{(i-1)}...n_0$ intr-o anumita baza *b*, scrierea polinomiala desfasurata este:

$$(N)_b=n_i\times b^i+n_{(i-1)}\times b^{(i-1)}+ n_{(i-2)}\times b^{(i-2)}+...+ n_0\times b^0$$

Exemplul 1:

1. Scrierea numărului zecimal 1023 in forma desfasurata, ca mai sus.
2. Stabilirea valorile in baza 10 ale numerelor:
   i. $(1023)_5$
   ii. $(10111)_2$

iii. $(23AF)_{16}$

Solutia primei parti: Numarul este compus din 4 cifre cu b=10 si i=3. Astfel, folosind ecuatia de mai sus, avem $n_3 = 1$, $n_2 = 0$, $n_1 = 2$ si $n_0 = 3$. Numarul desfasurat este obtinut prin inlocuirea acestor valori pentru a obtine $(1023)_{10} = 1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

Solutia celei de-a doua parti:

1. Aici avem i=3 si b=5. Rezulta

$$(1023)_5 = 1 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 3 \times 5^0$$
$$= 125 + 0 + 10 + 3 = (138)_{10}$$

2. $$(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$$
$$= 16 + 0 + 4 + 2 + 1 = (23)_{10}$$

3. $$(23AF)_{16} = (2 \times 16^3) + (3 \times 16^2) + (A \times 16^1) + (F \times 16^0)$$
$$= (2 \times 16^3) + (3 \times 16^2) + (10 \times 16^1) + (15 \times 16^0)$$
$$= 8192 + 768 + 160 + 15$$
$$= (9135)_{10}$$

Numere cu parte întreagă şi fracţionară

Un numar, $(N)_b$, *cu parte întreagă şi fracţionară* este reprezentat astfel

$$(N)_b = (n_i\, n_{(i-1)} \ldots n_0 \cdot n_{-1}\, n_{-2} \ldots n_{-m})_b$$

cu doua parti: (1) partea întreaga $n_i\, n_{(i-1)} \ldots n_0$ si (2) partea fractionara $n_{-1}\, n_{-2} \ldots n_{-m}$. Indicii corespund localizarii unei anumite cifre in raport cu baza. Indicii părții întregi incep cu 0, in timp ce ai partii fractionare pornesc de la -1. Pentru baza 10, punctul este numit virgulă (punct) zecimal. Pentru baza 2, punctul mai este numit si punct binar.

Pentru un numar dat intr-o baza oarecare, b, echivalentul in zecimal este obtinut prin insumarea tuturor cifrelor numarului, inmultind pe fiecare cu b^i, unde i este pozitia cifrei.

Exemplul 2:
Să se gasească echivalentul zecimal pentru:

1. $(1023.21)_5$
2. $(10111.01)_2$

Solutie: Folosind metoda ştiută, se obține

$$(1023.21)_5 = 1 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 3 \times 5^0 + 2 \times 5^{-1} + 1 \times 5^{-2}$$
$$= 125 + 0 + 10 + 3 + 2/5 + 1/25$$
$$= (138.44)_{10}$$

Pentru partea a 2-a, se obţine

$$(10111.01)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$
$$= 16 + 0 + 4 + 2 + 1 + 0 + 1/4$$
$$= (23.25)_{10}$$

Bazele cele mai uzuale in proiectarea digitala si in programarea in limbaj masina sunt baza 2 (numere binare), baza 8 si baza 16 (numere hexazecimale). Vom discuta in continuare procesul conversiei intre aceste trei baze.

Bazele 8 si 16

Bazele 8 si 16 sunt folosite pentru a înlesni comunicarea dintre utilizator si calculator. Calculatoarele proceseaza informatia binara. Pe langa date, programele care dau comenzi calculatorului sunt de asemenea reprezentate in format binar. Atât lungimea mare, cât mai ales gradul mare de similitudine dintre două şiruri binare fac imposibila diferentierea acestora. Astfel, este dificil de distins intre două instructiuni diferite cand sunt in formatul binar. Introducerea bazelor 8 si 16 (hexazecimala) au simplificat acest proces. Conversia intre aceste trei baze este abordata in continuare.

Cifrele unui numar binar sunt numite biti. Un numar cu n biti este un numar binar care contine n biti. Cifra cea mai semnificativa este cea mai din stanga (pentru numerele binare, este numita bitul cel mai semnificativ). Similar, cifra cea mai putin semnificativa este cea mai din dreapta (in cazul numerelor binare este numita bitul cel mai putin semnificativ).

Pentru a converti un numar binar in baza 8:

1. Grupam numerele binare in seturi a cate 3 biti.
   Punctul de referința este punctul binar.
   Pentru partea intreaga, ne mutam de la punctul binar spre stanga; pentru cea fractionara pornim spre dreapta.
2. Daca este nevoie, adaugam zerouri la stanga partii intregi si la dreapta partii fractionare, pentru a forma grupuri a cate 3 biti.
3. Inlocuim fiecare grup format la punctele 1 si 2 cu echivalentul sau in baza 8.

Trebuie notat faptul ca adaugarea de zerouri la stanga partii intregi si/sau la dreapta celei fractionare nu schimba valoarea numarului initial.

Pentru a converti un numar binar in hexazecimal vom folosi procedura listata mai sus; totusi, biti sunt grupati cate 4.

Pentru a realiza punctul 3 de mai sus, vom folosi tabelul 1.4.1(a) si (b). Urmatoarele exemple ilustreaza folosirea procedurii subliniate mai sus.

Tabelul 1.4.1 (a)

Folosit pentru a converti un numar binar de 3 cifre in echivalentul sau in baza 8

Binar	Echivalent in baza 8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Tabelul 1.4.1 (a)

Folosit pentru a converti un numar binar de 4 cifre in echivalentul sau hexazecimal

Binar	Echivalent zecimal	Echivalent hexazecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9

Exemplul 3:

Convertiti numarul binar 11010110.00111 in baza 8.

Cand grupam cifrele binare cate 3, se observa ca avem nevoie sa adaugam un zero in partea stanga a partii intregi si unul partea drepata a partii zecimale. Numarul modificat, gruparea si echivalentul in baza 8 sunt date in figura 1.4.1.

Exemplul 4:

Convertiti numarul binar 111000110.001 in hexazecimal.

Similar, in gruparea cifrelor binare in grupe de cate 4, se observa ca avem nevoie sa adaugam trei zerouri la stanga partii intregi si un zero in dreapta partii fractionare.

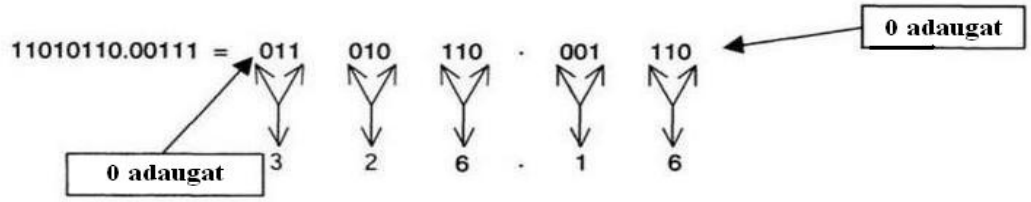

Figura 1.4.1

Conversie din binar in octal

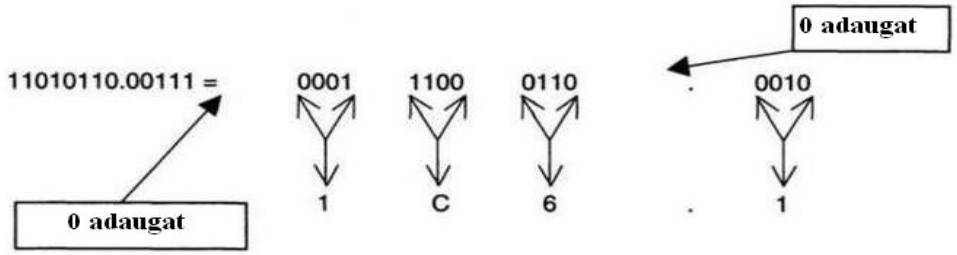

Figura 1.4.2

Conversie din binar in hexazecimal

Numarul modificat, gruparea si echivalentul hexazecimal sunt date in **figura 1.4.2.**

Inainte de a incheia aceasta sectiune va amintim ca procesul descris in diagramele de mai sus este reversibil, adica find dat un numar in forma octala sau hexazecimala, forma numarului in baza 2 poate fi gasita prin asocierea fiecarui grup secventa de biti necesara. Mai mult, trecerea de la baza 8 la baza 16 si invers poate fi facuta prin convertirea numarului in binar si apoi efectuand conversia din binar in celalalt format (ca in fig 1.4.1 sau 1.4.2).

Exemplul 5:

Dat find numarul octal $(127.25)_{8,}$ gasiti echivalentul sau hexazecimal.
Solutie: Prima data convertim numarul in binar. Bitii numerelor binare echivalente sunt apoi regrupati in grupe de cate 4 si numarul hexazecimal corespunzator este gasit.

$$(127.25)_8 \xrightarrow{\text{in binar}} \underbrace{001}_{1}\underbrace{010}_{2}\underbrace{111}_{7}.\underbrace{010}_{2}\underbrace{101}_{5} \xrightarrow{\text{in hex}} \underbrace{0000}_{0}\underbrace{0101}_{5}\underbrace{0111}_{7}.\underbrace{0101}_{5}\underbrace{0100}_{4}$$

$$= (57.54)_{16}$$

Tipuri de operanzi si mulțimile de definiție

Vom începe acest paragraf prin considerarea tipurilor de date folosite in limbajele de programare si matematica.

Tabelul 1.5.1

Tipuri de date

Tipuri de date din calculator	Echivalente matematice	Exemple
Fara semn	Numere naturale	0 , 5 , 7
Cu semn	Numere intregi	-5 , 0 , 1 , 6
Virgula fixa	Numere rationale	1.2 , 1.5
Virgula mobila	Numere rationale	$-2.1x10^5$
Caracter	-	‚A’

Tipuri de date

In matematica numerele sunt grupate in functie de setul lor de valori posibile. In particular, aceasta aranjare include seturile de numere naturale, numere intregi, numere rationale, numere reale si numere complexe,multimi care sunt pastrate si in reprezentarea numerelor in calculator. Acest lucru este implementat intr-un limbaj de programare, spre exemplu: in partea de declarare a variabilelor, care este extrem de importanta. Reprezentarea numerelor in calculator e exemplificata in tabelul 1.5.1
In aritmetica unui calculator, cu exceptia ultimului rand, se lucreaza cu titpurile de date din coloana 1 a tabelului 1.5.1. Numerele fara semn sunt defapt intregi nenegativi (numerele nu sunt asociate cu un semn); numerele cu semn reprezinta numere intregi cu sau far semn. Cele doua reprezentari ramase sunt folosite pentru a aproxima numerele reale. Reprezentarile in virgula fixa conti intregi cu sau fara semn cu un punct care separa 2 parti : partea intreaga si partea fractionala. Reprezentarea in virgula mobila este conpusa tot din doua parti : o parte fixa cu punct si un exponent. De exemplu, $-2.1x10^5$ contine 2 parti: -2.1 este partea fixa cu punct si 10^5 este exponentul. Vom vorbi mai multe despre aceste reprezentari in capitolele ce urmeaza.

Limite finite

In matematica se folosesc multimi cu cardinalul infinit (cardinalul unei multimi este numarul de elemente a multimii). Calculatoarele contin insa, elemente de stocare finite. Ca o urmare a acestui fapt, calculatoarele proceseaza doar submultimi ale acesto multimi finite. Cand se efectueaza operatii aritmetice cu opernazi (numere), acestia sunt stocati in registrii cu un numar finit de elemente de stocare. Un registru e caracterizat de numarul de biti pe care il contine. Un registru n-bit contine n elemente de stocare, fiecare putand retine un singur bit dintr-un total de n biti. Numarul finit de biti micsoreaza limitele multimilor de numere care pot fi stocate in registrii.
In continuare ne vom referi in discutia noastra la kilometrajul unei masini care retine cifre zecimale. Numarul de cifre din aparatul de kilometraj este finit:

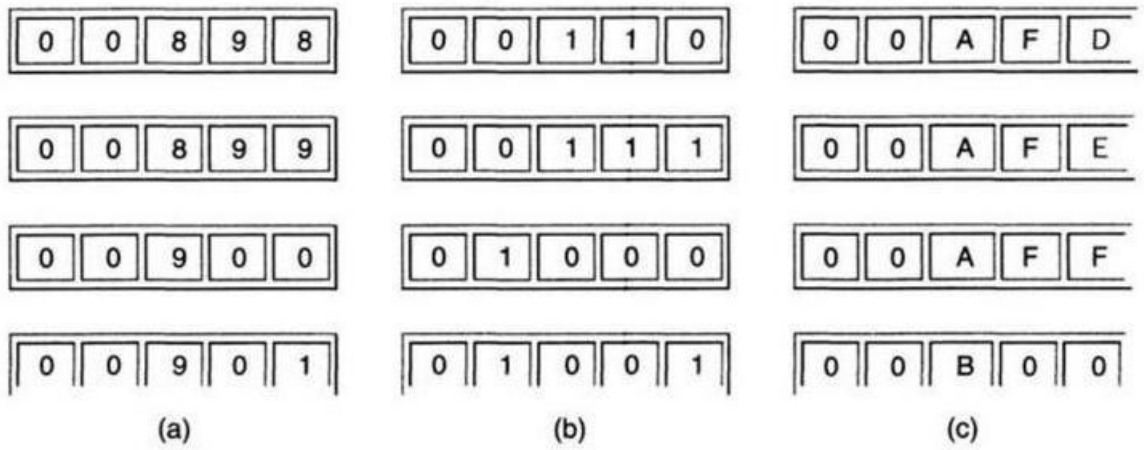

Figura 1.5.1 Numararea in baze diferite

Ca urmare, multimea de numere care poate fi stocata (inregistrata) este deasemenea finita. Pentru un aparat de kilometraj cu 5 cifre zecimale, multimea numerelor care pot fi inregistrate este de la cel mai mic 00000 pana la cel mai mare $99999=10^5-1$. Aceasta inseamna un total de 10^5 numere. Numerele stocate sunt fara semn. Analog kilometrajelor, numerele in calculatoare sunt stocate in registrii de dimensiune finita.

Vom asemana limitele multimilor de numere in baze diferite cu procesul de a numara in baza 10. De aici putem deduce ecuatii pentru cel mai mic sic el mai mare numar intr-o baza data.

La numarare, intotdeauna ultima cifra cea mai nesemnificativa se schimba. Aceasta cifra este incrementata pana ajunge la valorea ei maxima (9 in cazul bazei 10). De aici rezulta ca valoarea maxima este egala cu b-1 unde b este o baza oarecare . Urmatoarea numarare face ca cea mai nesemnificativa cifra sa devina 0. Cifra urmatoare insa, se schimba si ea. In general, pentru ca o anumita cifra sa se schimbe (sa fie incrementata sau sa devina resetata la 0), toate cifrele predecesoare trebuie sa aibe valoarea maxima b-1. In baza 10, toate cifrele predecesoare trebuie sa fie de valoare 9. In baza 2, aceasta se intampla cand toate cifrele precedente au valoarea 2-1=1 ca in figura 1.5.1 b) randurile 2 si 3. In hexazecimal, acest lucru se intampla cand toate cifrele zecimale predecesoare au valoarea F, ca in figura 1.5.1 c), randurile 3 si 4.

In figura 1.5.1 a) intervalul valorilor care pot fi stocate este de la 0 la 99999 (10^5-1) din moment ce numarul de cifre folosite este 5. Inregistrand valori in afara acestui interval cauzeaza o eroare numita ‚overflow' daca numarul e peste 99999 sau ‚underflow' daca numarul e sub 0.

In caz general, cel mai mare numar fara semn pe care il putem stoca intr-un registru n-bit care stocheaza cifre in baza b este b^n-1. Pentru a numara in baza 2 folosind un registru n-bit, cel mai mare intreg fara semn care poate fi stocat este 2^n-1. De exemplu, pentru registrii de 3-, 4-, si 5- biti, se pot inregistra intregi binari fara semn intre $(000)_2$ si $(111)_2=7$, $(0000)_2$ si $(1111)_2=15$, respectiv $(00000)_2$ si $(11111)_2=31$.

Intervalul numerelor se schimba daca numarul care este stocat in format virgula mobila. Pentru acest format trebuie sa includem si locatia virgulei care este de obicei la extremitatea dreapta a registrului (pentru numere intregi) sau la extremitatea stanga a registrului (pentru numere fractionale). Daca continutul unui registru n-bit reprezinta o fractie, atunci cel mai mic numar este zero (toti bitii sunt zero) si cel mai mare numar este:

$(a_{-1}a_{-2}a_{-3}a_{-4}...a_{-m})_2=(.1111...1)_2$,unde cifra 1 apare de m ori

$=2^{-}1+2^{-2}+2^{-3}+...+2^{-m}$

$=2^{-m}(2^{-m-1}+2^{m-2}+2^{m-3}+...+2^{0})$
$=(2^{m}-1)/\ 2^{m}$
$=1-2^{-m}$

Din acest motiv, numarul de 3 biti $(111)_2$ poate fi interpretat ca un decimal fara semn (7) sau un fractional (7/8).

Inainte de a trece la sectiunea urmatoare, vom lista cele mai utilizate unitati de calcul binar: 1 K (kilo)=2^{10}=1024 (aproximativ 1000); 1M (mega)=2^{20}; 1G (giga)=2^{30}. Marimea unui registru poate fi data in octeti (1 octet=8 biti). De aici, un registru de 4 octeti are 32 de biti. Deasemenea, marimea unui registru este deseori numita un cuvant. Rezulta ca un registru de 32 biti are marimea unui cuvant de 32 de biti. Cel mai mare intreg fara semn care poate fi stocat in acest registru este de 2^{32}-1 care este de aproximativ 4 G.

Conversia numerelor decimale in numere echivalente din baze arbitrare

Dat fiind numarul decimal $(N)_{10}=(n_i n_{(i-1)}...n_0.n_{-1}n_{-2}...n_{-m})_{10}$; pentru a gasi echivalentul sau intr-o baza diferita, folosim doua metode aplicate partii intregi si fractionale ale lui N.

Conversia partii intregi

Procesul de conversie a unui numar decimal fara parte fractionara intr-un numar echivalent intr-o baza oarecare b, este dat in algoritmul 1.

> **Algoritmul 1:** Dat fiind un numar in baza 10, N, fara parte fractionala, echivalentul sau intr-o baza oarecare b poate fi obtinut prin impartiri repetate a numarului original si a tuturor caturilor rezultate prin baza b. Resturile sunt salvate in ordinea in care sunt formate. Procesul se termina cand ultimul cat obtinut este 0. Numarul echivalent (in baza b) este obtinut prin listarea resturilor de la cea mai putin semnificativa cifra pana la cea mai semnificativa cifra in ordinea in care au fost formate resturile.

TABELUL 1.6.1

Conversia din baza 10 intr-o baza arbitrara

Resturile r_i	Numarul inintial (N) si caturile succesive(q_i)
	N=138
r_0=3	q_0=27
r_1=2	q_1=5
r_2=0	q_2=1
r_3=1	q_3=0

Exemplul de mai jos ilustreaza metoda evidentiata mai sus.

Exemplul 6:
Convertiti numarul $(138)_{10}$ in echivalentul sau in baza 5.
Aplicam procedura descrisa mai devreme, la fel ca in tabelul **1.6.1** . In acest exemplu, numarul initial 138 este afisat in partea dreapta sus a tabelului. Restul diviziunii a lui 138 cu 5 este r_0=3. Catul (q_0) este 27 ca in tabel. Randul 3 al tabelului este obtinut prin repetarea procesului de impartire, insa de aceasta data prin folosirea catului 27 in loc de numarul initial. In final, ultimul rand se obtine prin impartirea catului 1 (q_2=1) la 5, salvandu-se restul corespunzator si catul in acest rand. Cum noul cat este 0, procesul de conversie se opreste. Resturile obtinute (scrise in ordinea formarii de la dreapta la stanga) constituie numarul echivalent in baza 5, adica : $(138)_{10}=(1023)_5$

Exemplul 7:
Convertiti numarul 23 (in baza 10) intr-un numar echivalent in baza 2. Folosind procedura evidentiata in algoritmul 1, obtinem rezultatele afisate in figura 1.6.1.

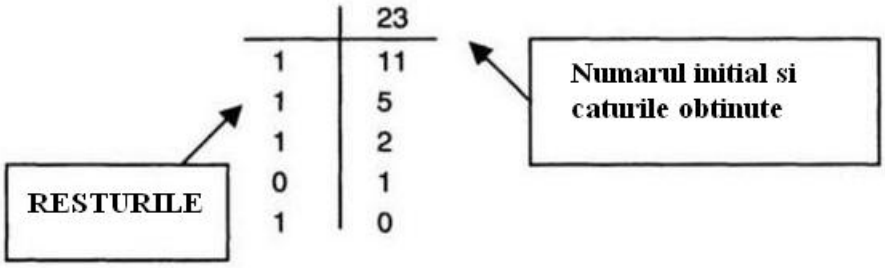

Figura 1.6.1

Conversia din baza 10 in baza 2

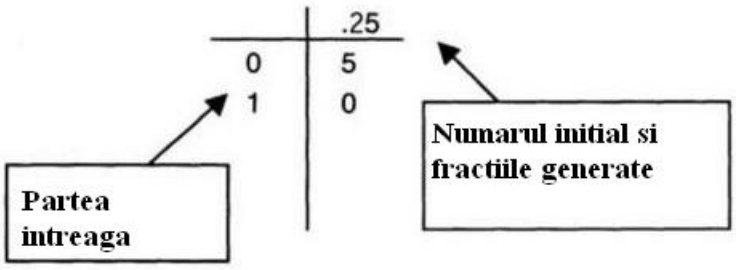

Figura 1.6.2

Conversia din baza 10 in baza 2 (partea fractionara)

Reprezentarea binara a numarului 23 este : $(23)_{10}=(10111)_2$

Conversia părții fracționare

Procesul de conversie a partii fractionare a unui numar in baza 10 intr-un numar echivalent intr-o baza arbitrara b este dat in algoritmul 2.

> **Algoritmul 2:** Se inmulteste in mod repetat partea fractionala si toate fractiile generate ulterior cu baza b, rezultatele intregi din fiecare multiplicare fiind salvate in ordinea formarii. Se continua pana partea fractionala este 0. Echivalentul fractional in baza b se obtine scriind cifrele intregi (in ordinea formata) de la cea mai semnificativa (in stanga) la cea mai putin semnificativa (in dreapta), apoi adaugandu-se punctul in partea stanga a numarului.

Exemplul de mai jos ilustreaza folosirea algoritmului 2.

Exemplul 8:
Convertiți fracția zecimala 0.25 in fractia echivalenta in baza 2.

Aplicand algoritmul 2, obtinem rezultatele din figura 1.6.2. Fractia initiala 0.25 este scrisa in randul 1. Randul 2 contine 2 coloane, coloanele combinate sunt rezultatul operatiei 2*(2.5). Multiplicand fractia 5 din randul 2 cu numarul 2, obtinem rezultatul afisat in randul 3. Cum fractia noua este 0 algoritmul se termina aici.

Numarul fractional echivalent in baza 2 este obtinut prin scrierea partilor intregi de la stanga la dreapta in ordinea in care au fost formate si adaugand virgula la stanga numarului. Relulta ca: $(0.25)_2=(0.01)_2$

Exemplul 9:
Convertiti fractia zecimala $(0.44)_{10}$ intr-o fractie echivalenta in baza 5.

Urmand procedura descrisa in algoritmul 2, obtinem rezultatele din figura 1.6.3. Obtinem $(0.44)_{10}=(0.21)_5$

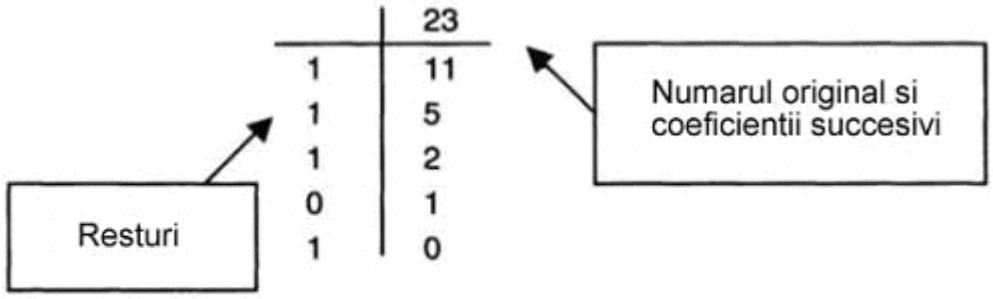

FIGURA 1.6.1
Conversia din baza 10 in binar (partea intreaga)

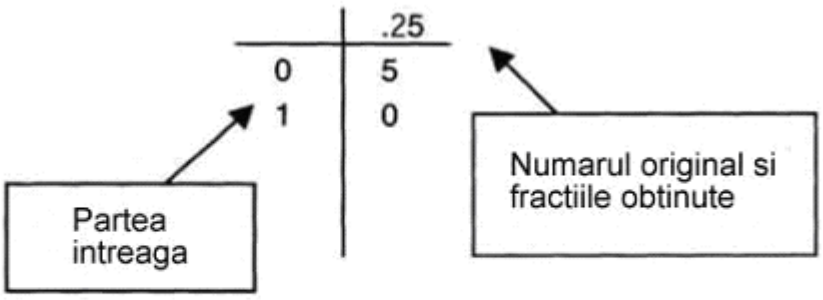

FIGURA 1.6.2

Conversia din baza 10 in binar (partea fractionara)

Exemplu 10:

Transformam $(0.7)_{10}$ intr-o fractie binara echivalenta.

Figura 1.6.4 arata procesul de conversie. Dupa cum s-a stabilit mai devreme, rezultatul din linia 3 (0.8) este obtinut prin inmultirea partii fractionare din linia precedenta (0.4) cu 2. Ultima linie are partea fractionara 0.4. Ca urmare, inmultind repetat cu 2, secventa de cifre obtinute este (0110). Aceasta secventa este repetata la infinit, deoarece criteriu de oprire al algoritmului (partea fractionara sa fie 0) nu poate fi satisfacut.

Exemplul anterior ne arata un lucru interesant despre conversia numerelor: anumite numere sunt memorata in calculator intr-o reprezentare aproximativa si nu exacta. (In matematica, ecuatia x*1/x=1 este tot timpul adevarata, cand x diferit de 0. Nu acelasi lucru se intampla cand testam conditia in limbajele de programare.)

Pentru a transforma un numar zecimal cu virgula fixa intr-un numar echivalent intr-o baza arbitrara, aplicam algoritmul asupra partii intregi si partii fractionare, dupa care combinam rezultatele.

Exemplu 11:

Transformam $(138.44)_{10}$ într-un număr echivalent în baza 5.

Din exemplele anterioare, avem ca $(138)_{10} = (1023)_5$, si $(0.44)_{10} = (0.21)_5$.
Combinând rezultatele, avem $(138.44)_{10} = (1023.21)_5$