2.6. Reprezentarea numerelor în virgulă mobilă

Multe aplicații necesită numere care nu sunt întregi. Există mai multe posibilități pentru reprezentarea acestor numere. O posibilitate este reprezentarea în *virgulă fixă*. În acest caz, se poate utiliza aritmetica pentru numere întregi, iar apoi se plasează virgula binară în poziția predefinită, de exemplu, după bitul de semn. Adunarea a două numere reprezentate într-un asemenea format poate fi realizată cu un sumator pentru numere întregi, în timp ce înmulțirea necesită operații suplimentare de deplasare. În cazul reprezentării numerelor în virgulă fixă, deşi virgula nu mai este reprezentată fizic în calculator, poziția virgulei binare, stabilită prin proiectare, nu mai poate fi schimbată. Pentru transformarea tuturor numerelor în acest format, trebuie executate o serie de operații de scalare sau deplasare, ataşând numerelor factori de scală. Evidența acestora trebuie realizată prin program, ceea ce măreşte timpul de calcul. Alte reprezentări care au fost propuse constau în păstrarea logaritmilor numerelor şi executarea înmulțirii prin adunarea logaritmilor, sau utilizarea unei perechi de întregi (x, y) pentru a reprezenta fracția x/y.

O soluție mai avantajoasă este utilizarea unei tehnici de scalare automată, cunoscută sub numele de reprezentare în *virgulă mobilă* (numită şi reprezentare în *virgulă flotantă* sau notație ştiințifică). În acest caz, factorul de scală devine o parte a cuvântului din calculator, poziția virgulei variind pentru fiecare număr în mod automat.

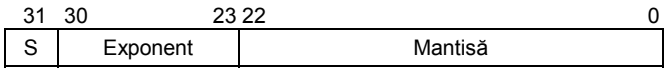
2.6.1. Principii

În general, un număr N se poate reprezenta în virgulă mobilă (VM) în forma următoare:

$$N = \pm M \cdot B^{\pm E} \tag{2.25}$$

Un număr reprezentat în VM are două componente. Prima componentă este *mantisa* (M), care indică valoarea exactă a numărului într-un anumit domeniu, fiind reprezentată de obicei ca un număr fracționar cu semn. A doua componentă este *exponentul* (E), care indică ordinul de mărime al numărului. În expresia de sus, B este baza exponentului.

Această reprezentare poate fi memorată într-un cuvânt binar cu trei câmpuri: semnul, mantisa şi exponentul. De exemplu, presupunând un cuvânt de 32 de biți, o asignare posibilă a biților la fiecare câmp poate fi următoarea:



Aceasta este o reprezentare în mărime şi semn, deoarece semnul are un câmp separat față de restul numărului. Câmpul de semn constă dintr-un bit care indică semnul numărului, 0 pentru un număr pozitiv şi 1 pentru un număr negativ. Nu există un câmp rezervat pentru baza B, deoarece această bază este implicită şi ea nu trebuie memorată, fiind aceeaşi pentru toate numerele.

De obicei, câmpul rezervat *exponentului* nu conține exponentul real, ci o valoare numită *caracteristică*, care se obține prin adunarea unui deplasament la exponent, astfel încât să rezulte întotdeauna o valoare pozitivă. Astfel, nu este necesar să se rezerve un câmp separat pentru semnul exponentului. Caracteristica C este deci exponentul deplasat:

$$C = E + deplasament \tag{2.26}$$

Valoarea reală a exponentului se poate afla prin scăderea deplasamentului din caracteristica numărului. De exemplu, dacă pentru caracteristică se rezervă un câmp de 8 biți, valorile caracteristicii pot fi cuprinse între 0 şi 255. Considerând un deplasament de 128 (80h), exponentul real poate lua valori între –128 şi +127, fiind negativ dacă $C < 128$, pozitiv dacă $C > 128$, şi zero dacă $C = 128$. Exponentul este deci reprezentat *în exces* 128.

Unul din avantajele utilizării exponentului deplasat constă în simplificarea operațiilor executate cu exponentul, datorită lipsei exponenților negativi. Al doilea avantaj se referă la modul de reprezentare al numărului zero. Mantisa numărului zero are cifre de 0 în toate pozițiile. Exponentul numărului zero poate avea, teoretic, orice valoare, rezultatul fiind tot zero. La unele calculatoare, dacă un rezultat are mantisa zero, exponentul rămâne la valoarea pe care o are în momentul respectiv, rezultând un "*zero impur*".

La majoritatea calculatoarelor, se recomandă ca numărul zero să aibă cel mai mic exponent posibil, rezultând astfel un "*zero pur*". În cazul exponenților deplasați, exponentul cu cea mai mică valoare este 0. Deci, prin utilizarea caracteristicii, reprezentarea în VM a numărului zero este aceeaşi cu reprezentarea în VF, adică toate pozițiile sunt 0. Aceasta înseamnă că se pot utiliza aceleaşi circuite pentru testarea valorii zero.

Un alt avantaj al utilizării exponenților deplasați este că numerele pozitive în virgulă mobilă sunt ordonate în acelaşi fel ca şi numerele întregi. Deci, mărimea numerelor în virgulă mobilă poate fi comparată utilizând un comparator pentru numere întregi.

Un dezavantaj al utilizării exponenților deplasați este că adunarea lor este mai complicată, deoarece necesită scăderea deplasamentului din suma exponenților.

În reprezentarea ilustrată anterior, *mantisa* constă din 23 de biți. Deşi virgula binară nu este reprezentată, se presupune că ea este aşezată înaintea bitului c.m.s. al mantisei. De exemplu, dacă $B = 2$, numărul 1,75 poate fi reprezentat sub mai multe forme:

$+0{,}111 \cdot 2^1$	0	1000 0001	1110 0000 0000 0000 0000 000
$+0{,}00111 \cdot 2^3$	0	1000 0011	0011 1000 0000 0000 0000 000

Pentru simplificarea operațiilor cu numere în VM şi pentru creşterea preciziei acestora, se utilizează reprezentarea sub forma normalizată. Un număr în VM este *normalizat* dacă bitul c.m.s. al mantisei este 1. Din cele două reprezentări ale numărului 1,75 ilustrate anterior, prima este cea normalizată.

Deoarece bitul c.m.s. al unui număr normalizat în VM este întotdeauna 1, acest bit nu este de obicei memorat, fiind un *bit ascuns* la dreapta virgulei binare. Aceasta permite ca mantisa să aibă un bit semnificativ în plus. Astfel, câmpul de 23 de biți este utilizat pentru memorarea unei mantise de 24 de biți cu valori cuprinse între 0,5 şi 1,0.

Cu această reprezentare, Figura 2.1 indică gama numerelor care pot fi reprezentate pe cuvânt de 32 de biți, în virgulă fixă şi în virgulă mobilă.

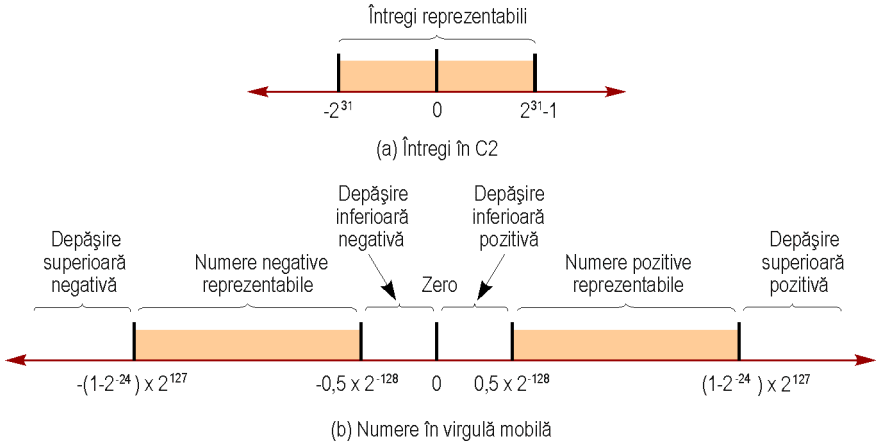


Figura 2.1. Numere care pot fi reprezentate în formate tipice de 32 de biți.

Dacă se utilizează reprezentarea în C2, se pot reprezenta toate numerele întregi între -2^{31} şi $2^{31}-1$, cu un total de 2^{32} numere diferite. Pentru formatul prezentat, se pot reprezenta numere în următoarele domenii (Figura 2.1):

- Numere negative între $-(1-2^{-24}) \cdot 2^{127}$ şi $-0,5 \cdot 2^{-128}$;
- Numere pozitive între $0,5 \cdot 2^{-128}$ şi $(1-2^{-24}) \cdot 2^{127}$.

Există cinci regiuni care nu sunt cuprinse în aceste domenii:

- Numere negative mai mici decât $-(1-2^{-24}) \cdot 2^{127}$, apariția acestora determinând o *depăşire superioară negativă*;
- Numere negative mai mari decât $-0,5 \cdot 2^{-128}$, care determină o *depăşire inferioară negativă*;
- Zero;
- Numere pozitive mai mici decât $0,5 \cdot 2^{-128}$, care determină o *depăşire inferioară pozitivă*;
- Numere pozitive mai mari decât $(1-2^{-24}) \cdot 2^{127}$, care determină o *depăşire superioară pozitivă*.

În unele cazuri, bitul ascuns se presupune poziționat la stânga virgulei binare. Astfel, mantisa memorată *M* va reprezenta de fapt valoarea 1,*M*. În acest caz, numărul normalizat 1,75 va avea următoarea formă:

$+1.11 \cdot 2^0$

0	1000 0000	1100 0000 0000 0000 0000 000

Presupunând că bitul ascuns este poziționat la stânga virgulei binare în formatul prezentat, un număr normalizat diferit de zero reprezintă următoarea valoare:

$$(-1)^S \cdot (1,M) \cdot 2^{E-128} \tag{2.27}$$

unde *S* indică bitul de semn.

În acest format se pot reprezenta numere în următoarele domenii:

- Numere negative între $-[1 + (1 - 2^{-23})] \cdot 2^{127}$ şi $-1,0 \cdot 2^{-128}$;
- Numere pozitive între $1,0 \cdot 2^{-128}$ şi $[1 + (1 - 2^{-23})] \cdot 2^{127}$.

Problema care apare în cazul formatului prezentat este că nu există o reprezentare pentru valoarea 0. Aceasta deoarece valoarea 0 nu poate fi normalizată. Totuşi, reprezentările în VM rezervă de obicei o combinație specială de biți pentru reprezentarea valorii 0. De multe ori, această valoare se reprezintă prin cifre de 0 în cadrul mantisei şi a exponentului. Un asemenea exemplu de reprezentare este formatul standard definit de organizația IEEE (standardul IEEE 754).

Depăşirea superioară apare atunci când exponentul depăşeşte valoarea maximă, de exemplu peste 127 în cazul formatului prezentat. *Depăşirea inferioară* apare atunci când exponentul are o valoare negativă prea mică, de exemplu sub –128. În cazul depăşirii inferioare, rezultatul se poate aproxima cu 0. Coprocesoarele matematice şi unitățile de calcul în virgulă mobilă au anumite mecanisme pentru detectarea, semnalarea şi tratarea depăşirii superioare şi a celei inferioare.

Pentru alegerea unui format în VM trebuie realizat un compromis între dimensiunea mantisei şi cea a exponentului. Creşterea dimensiunii mantisei va conduce la creşterea preciziei numerelor, iar creşterea dimensiunii exponentului va conduce la creşterea domeniului numerelor care pot fi reprezentate. Singura cale de a creşte atât precizia, cât şi domeniul numerelor, este de a utiliza un număr mai mare de biți pentru reprezentare. Cele mai multe calculatoare utilizează cel puțin două formate, în *simplă precizie* (de exemplu, pe 32 de biți), şi *dublă precizie* (de exemplu, pe 64 de biți).

2.6.2. Reprezentarea numerelor în formatul IEEE 754

În trecut, au existat diferențe considerabile în modul de execuție a operațiilor în VM la diferite familii de calculatoare. Aceste diferențe se refereau la numărul de biți alocați pentru exponent şi pentru mantisă, la gama exponenților, la modurile de rotunjire şi la operațiile executate la apariția unor condiții de excepție, ca depăşirea superioară sau cea inferioară. Pentru a facilita portabilitatea programelor de la un calculator la altul şi pentru a încuraja dezvoltarea programelor complexe orientate pe calcule numerice, *Societatea de Calculatoare* a organizației IEEE (*Institute of Electrical and Electronics*

Engineers) a elaborat un standard pentru reprezentarea numerelor în VM şi pentru operaţii aritmetice în această reprezentare. Acest standard a fost publicat în 1985.

Standardul era destinat în primul rând microprocesoarelor şi microcalculatoarelor, unde unii producători pun la dispoziţie doar posibilităţi limitate pentru calcule numerice. Ca rezultat al acestui standard, au fost dezvoltate circuite sau procesoare care implementează standardul. De exemplu, cele mai multe unităţi de calcul în VM şi coprocesoare matematice, printre care şi coprocesoarele matematice şi unităţile de calcul în VM ale procesoarelor *Intel* din familia 80x86, se conformează acestui standard.

Standardul IEEE 754 defineşte următoarele formate sau precizii: precizie *simplă*, precizie *simplă extinsă*, precizie *dublă* şi precizie *dublă extinsă*. Parametrii principali ai acestor formate sunt prezentaţi în Tabelul 2.7. Standardul nu precizează ca obligatorie implementarea tuturor formatelor, dar recomandă implementarea combinaţiei formatelor cu precizie simplă şi precizie simplă extinsă, sau a formatelor cu precizie simplă, precizie dublă şi precizie dublă extinsă.

Tabelul 2.7. Parametrii formatelor definite de standardul IEEE 754.

	Precizie simplă	**Precizie simplă extinsă**	**Precizie dublă**	**Precizie dublă extinsă**
Biţi ai mantisei	24	≥ 32	53	≥ 64
Exponent real maxim	127	≥ 1023	1023	≥ 16383
Exponent real minim	-126	≤ -1022	-1022	≤ -16382
Deplasament exponent	127	Nespecificat	1023	Nespecificat

Pentru toate formatele, baza implicită este 2. Formatele cu precizie simplă, precizie dublă şi precizie dublă extinsă sunt prezentate în Figura 2.2. Coprocesoarele matematice şi unităţile de calcul în virgulă mobilă ale procesoarelor implementează de obicei aceste formate.

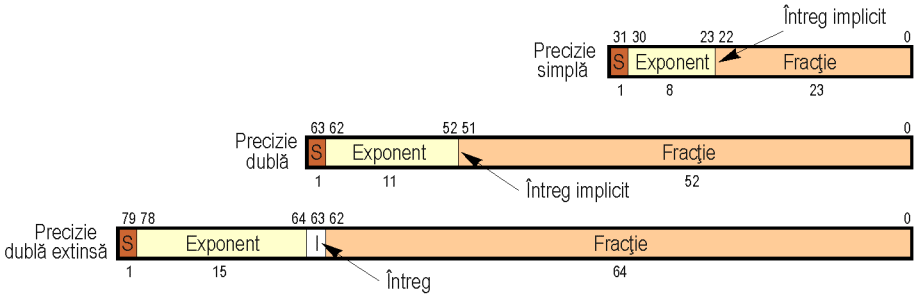


Figura 2.2. Formatele cu precizie simplă, precizie dublă şi precizie dublă extinsă definite de standardul IEEE 754.

S reprezintă semnul numărului. Pentru exponentul deplasat se rezervă 8 biţi în formatul scurt, 11 biţi în formatul lung şi 15 biţi în formatul temporar. Deplasamentul exponentului pentru cele trei formate este de 127 (7Fh), 1023 (3FFh), respectiv 16.383

(3FFFh). Valorile minime (0) şi cele maxime (255, 2047, respectiv 32.767) ale exponentului nu sunt utilizate pentru numerele normalizate, ele fiind utilizate pentru reprezentarea unor valori speciale.

Bitul ascuns este utilizat şi la standardul IEEE 754, dar mantisa este reprezentată într-un mod diferit. Reprezentarea mantisei este denumită *significand* în standardul IEEE. În cazul formatelor cu precizie simplă şi precizie dublă, mantisa constă dintr-un bit implicit cu valoarea 1 (partea întreagă), virgula binară implicită şi biţii fracţiei *F*:

$$M = 1,F$$

Dacă toţi biţii fracţiei sunt 0, mantisa este 1,0; dacă toţi biţii fracţiei sunt 1, mantisa este cu puţin mai mică decât 2,0. Deci:

$$1,0 \le M < 2,0$$

Formatul cu precizie dublă extinsă este utilizat pentru reprezentarea numerelor în cadrul unităţilor de calcul în VM şi a coprocesoarelor matematice, în scopul reducerii erorilor datorate rotunjirilor. În acest format, bitul 63 reprezintă partea întreagă a mantisei, care nu este implicită. Numerele în formatul temporar nu sunt întotdeauna normalizate, de aceea nu încep în mod obligatoriu cu un bit de 1. Din acest motiv, acest bit este reprezentat în mod explicit, fiind notat cu *I* în cadrul formatului. Valoarea mantisei este în acest caz:

$$M = I,F$$

Valoarea unui număr în precizie simplă (N_S), în precizie dublă (N_D) şi în precizie dublă extinsă (N_E) este:

$$N_S = (-1)^s \cdot M \cdot 2^{E-127} \tag{2.28}$$

$$N_D = (-1)^s \cdot M \cdot 2^{E-1023} \tag{2.29}$$

$$N_E = (-1)^s \cdot M \cdot 2^{E-16383} \tag{2.30}$$

Gama numerelor care pot fi reprezentate în precizie simplă este cuprinsă între aproximativ $1,18\times10^{-38}$ şi $3,4\times10^{38}$, cea a numerelor reprezentate în precizie dublă este cuprinsă între $2,23\times10^{-308}$ şi $1,79\times10^{308}$, iar cea a numerelor în precizie dublă extinsă este cuprinsă între $3,37\times10^{-4932}$ şi $1,18\times10^{4932}$.

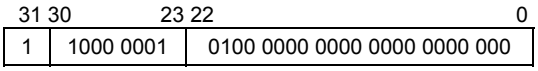
Exemple

1) Care este reprezentarea binară a numărului –0,75 în simplă precizie?

Numărul –0,75 poate fi scris ca –3/4 sau –0,11 în binar. Notaţia ştiinţifică a numărului este $-0,11\times2^0$, iar forma normalizată a acestei notaţii este $-1,1\times2^{-1}$. Exponentul va fi –1 + 127 = 126 (7Eh). Reprezentarea numărului în precizie simplă este deci:

31	30 … 23	22 … 0
1	0111 1110	1000 0000 0000 0000 0000 000

2) Care este numărul zecimal reprezentat de următorul cuvânt?



Bitul de semn este 1, câmpul rezervat exponentului conține `81h` = 129, iar câmpul fracției conține $1 \times 2^{-2} = 0{,}25$. Valoarea numărului este:

$$(-1)^1 \times 1{,}25 \times 2^{(129-127)} = -1{,}25 \times 2^2 = -1{,}25 \times 4 = -5{,}0$$

Una din problemele care apare la calculele cu numere în VM se referă la modul de tratare al depăşirilor inferioare şi superioare. O altă problemă este reprezentarea valorilor nedefinite. În acest scop, pe lângă numerele normalizate, standardul mai permite şi reprezentări ale unor valori speciale, pentru care sunt rezervate valoarea 0 şi valoarea maximă a exponentului. Valorile speciale pentru formatele cu precizie simplă şi precizie dublă sunt prezentate în Tabelul 2.8. În acest tabel se indică şi valorile numerelor normalizate în cele două formate.

Tabelul 2.8. Valori ale numerelor reprezentate conform standardului IEEE 754.

Precizie simplă (32 biți)			**Precizie dublă (64 biți)**		
Exponent	**Significand**	**Valoare**	**Exponent**	**Significand**	**Valoare**
0	0	$(-1)^S\ 0$	0	0	$(-1)^S\ 0$
0	≠0	$(-1)^S\ 2^{E-126}\ (0.F)$	0	≠0	$(-1)^S\ 2^{E-1022}\ (0.F)$
1…254	orice valoare	$(-1)^S\ 2^{E-127}\ (1.F)$	1…2046	orice valoare	$(-1)^S\ 2^{E-1023}\ (1.F)$
255	0	$(-1)^S \infty$	2047	0	$(-1)^S \infty$
255	≠0	NaN	2047	≠0	NaN

Pentru valoarea *zero*, atât exponentul, cât şi mantisa, sunt egale cu 0. Există două reprezentări pentru valoarea 0, în funcție de bitul de semn: +0, respectiv –0. Bitul ascuns de la stânga virgulei binare este implicit 0 în loc de 1.

În cazul obținerii unui rezultat cu o valoare mai mică decât numărul normalizat cel mai mic posibil, în mod obişnuit rezultatul este setat la zero şi calculele continuă, sau se semnalează o condiție de depăşire inferioară. Nici una din aceste soluții nu este satisfăcătoare. De aceea, standardul permite utilizarea numerelor care nu sunt normalizate, acestea fiind numite *numere denormalizate*. Caracteristica acestor numere este 0, iar mantisa este diferită de 0. În acest caz, bitul ascuns este 0.

Un număr denormalizat este generat printr-o tehnică numită *depăşire inferioară graduală*. Tabelul 2.9 prezintă un exemplu de aplicare a acestei tehnici în procesul de denormalizare.

Tabelul 2.9. Exemplu de aplicare a depăşirii inferioare graduale pentru denormalizare.

Operație	Semn	Exponent	Mantisă
Rezultat	0	–129	1,01011100000…00
Denormalizare	0	–128	0,10101110000…00
Denormalizare	0	–127	0,01010111000…00
Denormalizare	0	–126	0,00101011100…00
Rezultat denormalizat	0	–126	0,00101011100…00

Formatul utilizat este cel cu precizie simplă, astfel încât exponentul real minim este –126. Rezultatul din acest exemplu necesită un exponent egal cu –129 pentru a se obține un număr normalizat. Deoarece –129 este în afara domeniului permis pentru exponenți, rezultatul este denormalizat prin deplasarea mantisei la dreapta şi incrementarea exponentului până când acesta ajunge la valoarea minimă permisă –126. În cazul extrem, toți biții semnificativi sunt deplasați în afara mantisei, obținându-se un rezultat egal cu zero.

Pentru cazul în care apare o depăşire superioară, există o reprezentare specială pentru *infinit*, constând din exponentul cu valoarea maximă pentru formatul respectiv, şi mantisa egală cu 0. În funcție de bitul de semn, sunt posibile două reprezentări pentru infinit, +∞ şi –∞. Valoarea infinit se poate utiliza ca operand, utilizând reguli ca:

$$\infty + n = \infty$$
$$n / \infty = 0$$
$$n / 0 = \infty$$

Astfel, utilizatorul poate decide dacă va trata depăşirea superioară ca o condiție de eroare, sau va continua calculele cu valoarea infinit.

Pentru indicarea diferitelor condiții de excepție, ca în cazul operațiilor nedefinite de forma ∞/∞, ∞–∞, ∞ ∗ 0, 0/∞, 0/0, sau extragerea rădăcinii pătrate dintr-un număr negativ, s-a prevăzut un format special, care nu reprezintă un număr obişnuit, fiind numit NaN (*Not a Number*). Exponentul are valoarea maximă posibilă, iar mantisa este diferită de 0. Astfel, există o clasă întreagă de valori NaN.

Standardul IEEE specifică faptul că atunci când argumentul unei operații este NaN, rezultatul trebuie să fie NaN. Datorită regulilor de execuție a operațiilor aritmetice cu valori NaN, la scrierea subrutinelor de calcul în VM care acceptă o valoare NaN ca argument nu sunt necesare verificări speciale. De exemplu, presupunem că funcția *arccos* se calculează pe baza funcției *arctg*, utilizând formula:

$$arccos(x) = 2arctg\left(\sqrt{(1-x)/(1+x)}\right)$$

Dacă funcția *arctg* tratează un argument NaN în mod corect, şi funcția *arccos* va trata un asemenea argument în mod corect. Dacă x este o valoare NaN, $1+x$, $1–x$, $(1+x)(1–x)$ şi $\sqrt{(1-x)/(1+x)}$ vor fi de asemenea valori NaN. Astfel, nu este necesară testarea valorilor NaN.

O altă caracteristică a standardului IEEE cu implicații asupra circuitelor este regula de rotunjire. În urma operațiilor efectuate între două numere în VM, de obicei

rezultatul nu poate fi reprezentat în mod exact ca un alt număr în VM. Standardul specifică patru moduri de rotunjire: rotunjire spre 0, rotunjire spre +∞, rotunjire spre –∞, şi rotunjire la cel mai apropiat număr reprezentabil. Ultimul mod de rotunjire este cel implicit, şi este prevăzut pentru situaţiile în care numărul se află exact la jumătatea intervalului dintre două reprezentări în VM. Acest mod efectuează rotunjirea la un număr par.

Standardul IEEE defineşte cinci tipuri de excepţii: depăşire inferioară, depăşire superioară, împărţire la zero, rezultat inexact şi operaţie invalidă. În mod implicit, la apariţia unei asemenea excepţii, este setat un indicator şi calculele continuă. Standardul recomandă ca implementările să prevadă un bit de validare pentru fiecare excepţie. Dacă apare o excepţie cu bitul de validare setat, este apelată o rutină de tratare a excepţiei.

Excepţiile de depăşire inferioară, depăşire superioară şi împărţire la zero sunt prevăzute la majoritatea sistemelor de calcul în VM. Excepţia de *rezultat inexact* apare atunci când rezultatul unei operaţii trebuie rotunjit. Aceasta nu este o condiţie excepţională, deoarece apare în mod frecvent. Deci, validarea rutinei de tratare a acestei excepţii poate avea un impact semnificativ asupra performanţei. Excepţia de *operaţie invalidă* apare în cazul unor operaţii ca 0/0, ∞–∞ sau $\sqrt{-1}$.

Avantajul principal al standardului IEEE este că ajută la scrierea unor biblioteci de programe portabile. Acest standard are însă şi unele dezavantaje:

1. Standardul a fost destinat iniţial microprocesoarelor, astfel încât cerinţelor pentru implementările cu performanţe ridicate nu li s-a acordat o prioritate ridicată.
2. Standardul conţine specificaţii opţionale. Pentru cei care implementează standardul, este dificil să decidă care din aceste specificaţii vor fi implementate. Pentru programatorii care doresc scrierea unor programe portabile, problema este dacă trebuie să evite utilizarea specificaţiilor opţionale ale standardului.
3. Depăşirea inferioară graduală a fost implementată de obicei într-un mod care este cu mai multe ordine de mărime mai lent decât setarea rezultatului la zero, astfel încât de multe ori utilizatorii invalidează această tehnică.
4. Standardul nu descrie operaţiile aritmetice cu numere întregi şi nici funcţiile transcendentale (*sin*, *cos*, *exp*). În particular, standardul nu specifică acurateţea necesară pentru funcţiile transcendentale, sau valorile excepţionale ale funcţiilor transcendentale, ca de exemplu 0^0.

2.7. Coduri

2.7.1. Coduri binar-zecimale

Aceste coduri se utilizează pentru codificarea cifrelor zecimale. Pentru codificarea fiecăreia din cele 10 cifre, sunt necesari 4 biţi; din cele 16 valori posibile, 6 vor fi neutilizate. Prin stabilirea unor corespondenţe între mulţimea cifrelor zecimale şi