Operațiile aritmetice de bază pentru aceste numere sunt prezentate în Tabelul 6.12. Pentru adunare şi scădere este necesar ca ambii operanzi să aibă acelaşi exponent. Pentru aceasta poate fi necesară deplasarea virgulei binare a unuia din operanzi pentru a realiza alinierea mantiselor. Înmulțirea şi împărțirea nu necesită această operație.

Tabelul 6.12. Operații aritmetice cu numere reprezentate în virgulă mobilă.

Operație	Rezultat
$x + y$	$(M_x B^{E_x - E_y} + M_y) \times B^{E_y}, E_x \le E_y$
$x - y$	$(M_x B^{E_x - E_y} - M_y) \times B^{E_y}, E_x \le E_y$
$x \times y$	$(M_x \times M_y) \times B^{E_x + E_y}$
$x \div y$	$(M_x \div M_y) \times B^{E_x - E_y}$

6.3.1. Adunarea şi scăderea în virgulă mobilă

Adunarea şi scăderea în VM sunt mai complexe decât înmulțirea şi împărțirea. Aceasta deoarece pentru adunarea sau scăderea corectă a celor două numere, trebuie să se realizeze egalizarea exponenților acestora. Aceasta implică compararea mărimii exponenților şi apoi alinierea mantisei numărului cu exponentul mai mic. Algoritmul pentru adunare şi scădere are patru etape principale:

1. Alinierea mantiselor;
2. Adunarea sau scăderea mantiselor;
3. Normalizarea rezultatului;
4. Rotunjirea rezultatului.

Organigrama pentru adunare şi scădere este prezentată în Figura 6.17.

Înaintea operației, cei doi operanzi trebuie transferați în registrele care vor fi utilizate de UAL. Dacă formatul de reprezentare în VM cuprinde un bit implicit al mantisei, acest bit trebuie reprezentat în mod explicit pentru operație. În mod tipic, exponenții şi mantisele vor fi păstrate în registre separate, şi vor fi reunite după obținerea rezultatului.

Deoarece adunarea şi scăderea sunt identice cu excepția semnului diferit al operandului al doilea, în cazul operației de scădere se schimbă semnul scăzătorului. Următoarea etapă este alinierea mantiselor. Aceasta necesită compararea celor doi exponenți şi apoi alinierea mantisei numărului cu exponentul mai mic. Alinierea se realizează prin deplasarea repetată la dreapta a mantisei cu câte o poziție şi incrementarea exponentului până când cei doi exponenți devin egali. Dacă în urma acestui proces mantisa care a fost deplasată devine 0, atunci se raportează ca rezultat celălalt număr. Astfel,

dacă cele două numere au exponenţi care diferă în mod semnificativ, numărul mai mic va fi neglijat.

În continuare, se adună mantisele, ţinând cont de semnele acestora. Deoarece semnele pot fi diferite, rezultatul poate fi 0. Există de asemenea posibilitatea apariţiei unei depăşiri cu un bit a mantisei rezultatului. În acest caz, mantisa rezultatului este deplasată la dreapta cu o poziţie şi exponentul este incrementat. În urma deplasării, poate apare o depăşire a exponentului; în acest caz, se raportează o depăşire, iar operaţia este oprită.

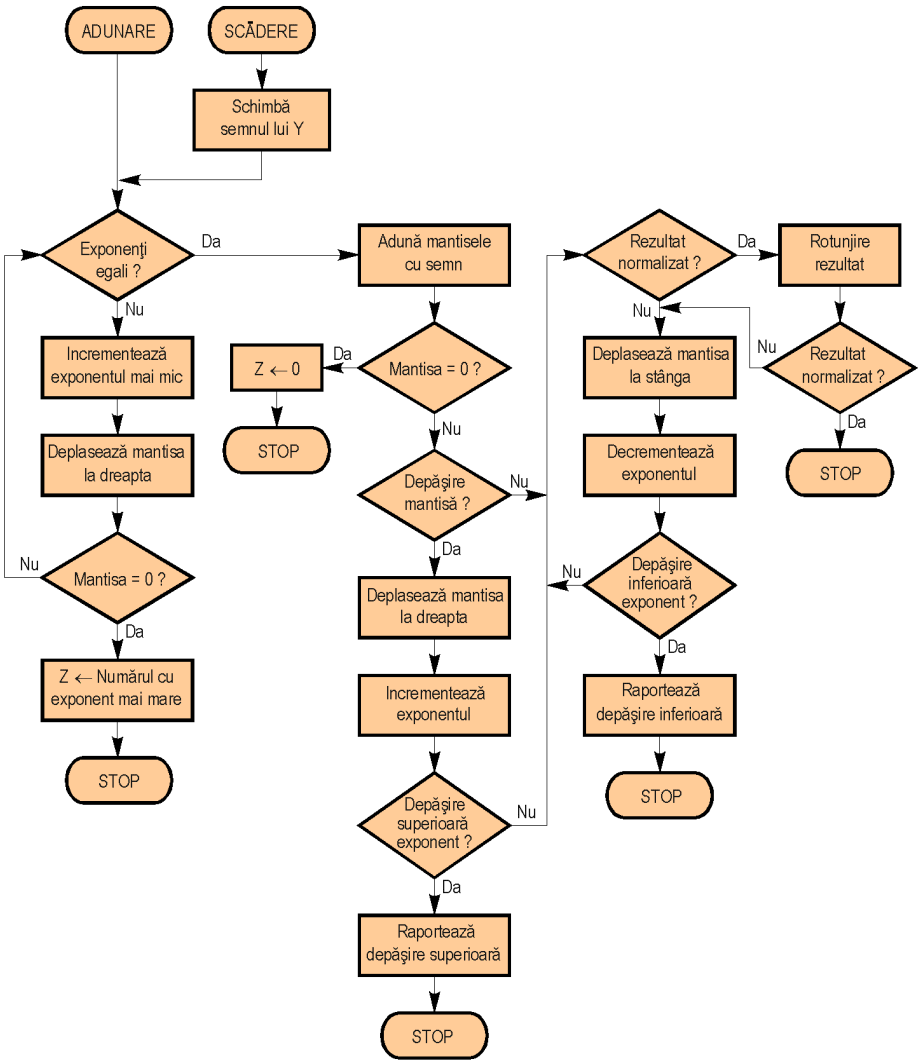

Figura 6.17. Adunarea şi scăderea în virgulă mobilă ($z \leftarrow x \pm y$).

În următoarea etapă se normalizează rezultatul. Normalizarea constă în deplasarea mantisei rezultatului la stânga până când bitul c.m.s. devine 1. După fiecare deplasare, se decrementează exponentul, ceea ce poate determina o depăşire inferioară a exponentului. În final, rezultatul este rotunjit. Procedura de rotunjire este prezentată în secţiunea 6.3.3. Dacă în urma rotunjirii rezultatul nu mai este normalizat, procedura de normalizare este repetată.

Exemplul 6.9

Să se adune numerele $0,5_{10}$ şi $-0,4375_{10}$ în binar utilizând algoritmul din Figura 6.17. Se presupune că precizia mantisei este de 4 biţi, exponentul real este cuprins între –126 şi 127, iar deplasamentul exponentului este 127.

Soluţie

Numerele scrise în binar utilizând notaţia ştiinţifică sunt:

$0,5_{10} = 0,1 = 0,1 \times 2^0 = 1,000 \times 2^{-1}$
$-0,4375_{10} = -0,0111 = -0,0111 \times 2^0 = -1,110 \times 2^{-2}$

Etapele algoritmului sunt următoarele:

1. Se deplasează la dreapta mantisa numărului mai mic ($-1,11 \times 2^{-2}$) şi se incrementează exponentul până când acesta devine egal cu exponentul numărului mai mare:

$$-1,110 \times 2^{-2} = -0,111 \times 2^{-1}$$

2. Se adună mantisele:

$$1,0 \times 2^{-1} + (-0,111 \times 2^{-1}) = 0,001 \times 2^{-1}$$

3. Se normalizează suma, verificând dacă apare depăşire superioară sau inferioară:

$$0,001 \times 2^{-1} = 0,010 \times 2^{-2} = 0,100 \times 2^{-3} = 1,000 \times 2^{-4}$$

Deoarece $-126 \le -4 \le 127$, nu există depăşire superioară sau inferioară (exponentul deplasat este $-4 + 127 = 123$, care este cuprins între 0 şi 255).

4. Se rotunjeşte suma:

$$1,000 \times 2^{-4}$$

Suma se poate exprima pe 4 biţi, astfel încât rotunjirea nu este necesară. Suma este deci:

$$1,000 \times 2^{-4} = 0,0001 = 1/16_{10} = 0,0625_{10}$$

Aceasta este suma corectă dintre $0,5_{10}$ şi $-0,4375_{10}$.

Multe calculatoare sau procesoare dispun de circuite dedicate pentru creşterea vitezei operațiilor în VM. Figura 6.18 prezintă o schemă bloc a unui circuit de adunare în VM.

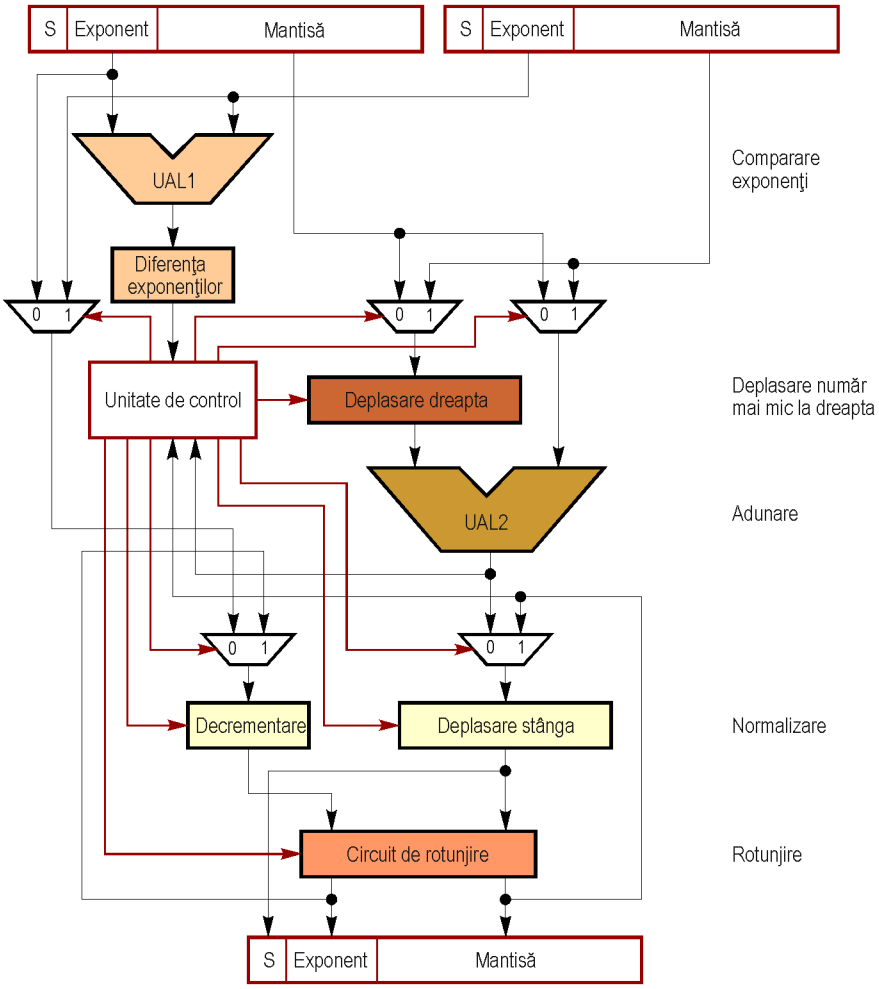

Figura 6.18. Schema bloc a unui circuit de adunare în virgulă mobilă.

Mai întâi se scade exponentul unui operand din exponentul celuilalt operand utilizând UAL1 pentru a determina care este exponentul mai mare. Diferența dintre exponenți controlează cele trei multiplexoare. Acestea selectează (de la stânga la dreapta) exponentul mai mare, mantisa numărului mai mic, respectiv mantisa numărului mai mare. Mantisa numărului mai mic se deplasează la dreapta, iar apoi se adună mantisele utilizând UAL2. Pentru normalizare se deplasează apoi suma la stânga şi se decrementează exponentul. Circuitul de rotunjire generează rezultatul final.