

## IV.4.7.



Depășiri pentru operații cu  
reprezentări în virgulă fixă

# Trecerea la reprezentări mai lungi

- Adăugare de cifre nesemnificative *la partea întreagă*
- Ce înseamnă cifră nesemnificativă ?
  - A+S: zerouri imediat după cifra semn
    - »  $\text{Val}_{A+S}^{8,0}(\text{00110011}) = 51$
    - »  $\text{Val}_{A+S}^{16,0}(\text{000000000110011}) = 51$
    - »  $\text{Val}_{A+S}^{8,0}(\text{10110011}) = -51$
    - »  $\text{Val}_{A+S}^{16,0}(\text{100000000110011}) = -51$
  - C<sub>1</sub>, C<sub>2</sub>: repetări ale cifrei semn imediat după ea
    - »  $\text{Val}_{C_2}^{8,0}(\text{0110011}) = 51 = \text{Val}_{C_2}^{16,0}(\text{00000000110011})$
    - »  $\text{Val}_{C_2}^{8,0}(\text{10110011}) = -77$
    - »  $\text{Val}_{C_2}^{16,0}(\text{111111110110011}) = -77$
    - »  $\text{Val}_{C_1}^{8,0}(\text{10110011}) = -76$
    - »  $\text{Val}_{C_1}^{16,0}(\text{111111110110011}) = -76$

- Interesează mai mult trecerea la reprezentări mai scurte: încapă reprezentarea corectă a rezultatului pe  $n+m$  biți?

# Mulțimea numerelor reprezentabile

- Submulțime finită a mulțimii numerelor raționale
- $Q_{\text{rep}}^{n,m}$ , unde  $\text{rep} \in \{A+S, C_1, C_2\}$
- $Q_{\text{rep}}^{n,m} \subset [\mathbf{min}; \mathbf{max}]$  
- $Q_{C_2}^{8,0} = \{-128; -127; -126; \dots -1; 0; 1; \dots 127\}$
- $Q_{C_2}^{4,4} = \{-8; -7,9375; \dots -0,0625; 0; 0,0625; \dots 7,9375\}$
- $Q_{A+S}^{n,m} = \{-(2^{n-1}-2^{-m}); -(2^{n-1}-2^{-m+1}); -(2^{n-1}-3 \times 2^{-m}); \dots -2^{-m}; 0; +2^{-m}; \dots (2^{n-1}-2^{-m})\}$
- $Q_{\text{rep}}^{n,m} \subset Q_{\text{rep}}^{n+k,m}$ ,  $k=1,2,\dots$  

# Importanța reprezentării în complement față de 2

- Cel mai simplu circuit de adunare a două reprezentări: sumator complet la fiecare rang și ignorarea transportului de la rangul cel mai semnificativ
  - **Temă:** adunarea în  $C_1$  implică o operație suplimentară
- Testare simplă a depășirilor
- Un număr în plus reprezentat pe aceeași lungime (reprezentare neredundantă)

# Operații în $C_2$

- Vom prezenta pe scurt doar proprietăți ale adunării și scăderii (suma algebrică) în  $C_2$
- Termenii sumei și rezultatul se reprezintă pe același număr de biți
  - Uneori această restricție cauzează depășiri
- Înmulțirea și împărțirea ar trebui să aibă, pentru reprezentarea rezultatului, o lungime dublă față de lungimea comună a operanzilor
  - Aceeași definiție a depășirii

# Depășiri

- Fie *rep* o reprezentare și *op* o operație cu numere. Fie  $a, b \in Q_{\text{rep}}^{n,m}$
- Spunem că operația *op* aplicată *numerelor* *a* și *b* produce depășire dacă:

$$a \text{ op } b \notin [\text{min}; \text{max}] \quad \leftarrow$$

- Este una din situațiile  **$\text{repr}(n1) \text{ op } \text{repr}(n2) \neq \text{repr}(n1 \text{ op } n2)$**
- Anume, eroare la partea întreagă
- Depășire înseamnă **rezultat eronat**
- Nu se poate produce depășire la partea fracționară
  - ci doar *aproximare* (rezultat aproximativ, nu eronat)

# Depășire - exemplul 1

- În reprezentarea  $C_2$ , cu  $n = 4$ ,  $m = 0$ :

$$1111 + 1111 = 11110 \rightarrow 1110$$

» Se "pierde" o cifră (rezultatul nu "încapă" pe 4 biți)

$$\text{val}_{C_2}^{4,0}(1111) = -1$$

$$(-1) + (-1) = -2$$

$$\text{val}_{C_2}^{4,0}(1110) = -2$$

» Rezultat corect, deci **nu se produce depășire**

» Testarea depășirii cu definiția:

$$\text{» } Q_{C_2}^{4,0} \subset [-8; 7] \quad -1 \in Q_{C_2}^{4,0} \quad -2 \in [-8; 7]$$

» Calculatorul nu poate însă testa definiția, întrucât "știe" reprezentările, nu numerele reprezentate



## Depășire - exemplul 2

- În reprezentarea  $C_2$ , cu  $n = 4$ ,  $m = 0$  :

$$0111 + 0111 = 1110$$

» Nu se "pierde" nici o cifră

$$\text{val}_{C_2}^{4,0}(0111) = 7$$

$$7 + 7 = 14$$

$$\text{val}_{C_2}^{4,0}(1110) = -2$$

» Rezultat eronat, deci **se produce depășire**

» Testarea depășirii cu definiția:

$$\text{» } Q_{C_2}^{4,0} \subset [-8; 7] \quad 7 \in Q_{C_2}^{4,0} \quad 14 \notin [-8; 7]$$

» Calculatorul nu poate însă testa definiția, întrucât nu "știe" numerele reprezentate

## Depășirea la adunare în $C_2$

- De fapt, cifra de transport de la rangul cel mai semnificativ (fie 0, fie 1) se pierde întotdeauna din rezultat
  - Dar se reține în indicatorul C (transport general)
- Trebuie găsită o condiție *asupra reprezentărilor* care, testată, să semnaleze depășirea
  - Fără a folosi definiția - numerele nu sunt disponibile
  - Testarea va fi făcută de un circuit atașat sumatorului

# Condiția de depășire

- Un alt mod de a vedea depășirea la adunare:
  - Depășire se produce numai atunci când ambii operanzi au același semn, iar reprezentarea rezultatului indică semn opus.
- Temă:
  - Nu este posibil ca definiția depășirii (cu numere) să fie satisfăcută când numerele au semn opus.
  - Studiul adunării reprezentărilor pentru cazurile: ambele numere pozitive, ambele numere negative. Concluzie.

# Rezultate asupra sumeii algebrice în $C_2$

- **Teorema 1.**

Dacă  $a, b \in Q_{C_2}^{n,m}$ , atunci  $a \pm b \in Q_{C_2}^{n+1,m}$

- **Lema 2.**

Dacă

$a = \text{Val}_{C_2}^{n+1,m}(\alpha_n \alpha_{n-1} \dots \alpha_1 \alpha_0 \alpha_{-1} \dots \alpha_{-m})$ , cu  $\alpha_n = \alpha_{n-1}$

atunci

$a = \text{Val}_{C_2}^{n,m}(\alpha_{n-1} \alpha_{n-2} \dots \alpha_1 \alpha_0 \alpha_{-1} \dots \alpha_{-m})$

# Rezultate asupra sumei algebrice în $C_2$

- **Definiție.** Date reprezentările  $\alpha$  și  $\beta$ ,

$$\alpha = \alpha_{n-1} \dots \alpha_1 \alpha_0 \alpha_{-1} \dots \alpha_{-m} \quad \text{și}$$

$$\beta = \beta_{n-1} \dots \beta_1 \beta_0 \beta_{-1} \dots \beta_{-m} \quad (\alpha_i, \beta_i \in \{0, 1\}),$$

suma lor formală  $\gamma = \alpha + \beta$  este

$$\gamma = \gamma_n \gamma_{n-1} \dots \gamma_1 \gamma_0 \gamma_{-1} \dots \gamma_{-m}, \text{ unde}$$

$$\sum_{i=-m}^n (\gamma_i \times 2^i) = \sum_{i=-m}^{n-1} ((\alpha_i + \beta_i) \times 2^i)$$

- **Teorema 3.** Dacă suma algebrică a numerelor reprezentate de  $\alpha$  și  $\beta$  nu produce depășire, atunci reprezentarea rezultatului este

$$\gamma_{n-1} \dots \gamma_1 \gamma_0 \gamma_{-1} \dots \gamma_{-m}$$

- **Consecință:** sumatorul serial efectuează adunări corecte

- semnul se adună ca oricare altă cifră

# Testarea depășirii

- **Teorema 4.**
- Suma algebrică a numerelor reprezentate de  $\alpha$  și  $\beta$  nu produce depășire dacă cifrele transport  $C_{n-1}$  și  $C_n$  coincid.
  - $1111 + 1111$  : cifrele-transport  $C_3$  și  $C_4$  sunt identice (egale cu 1)
  - $0111 + 0111$  : cifrele-transport  $C_3$  și  $C_4$  diferă (prima este 1, a doua este 0)
- **Consecință:** testarea depășirii în UAL prin adăugarea la sumatorul serial a unei porți NXOR
  - în care intră cifrele-transport  $C_{n-1}$  și  $C_n$
  - și din care iese bitul-flag **O** (Overflow)
  - cifra-transport  $C_n$  poziționează și bitul-flag **C** (carry)

IV.4.8.

Reprezentarea numerelor reale:  
aritmetica în virgulă mobilă

# Reprezentarea în virgulă mobilă

- Reprezentările în virgulă fixă nu sunt potrivite pentru numere reale
- Dualitatea magnitudine / precizie
  - lungimea  $n+m$  fixată
  - creșterea magnitudinii înrăutățește precizia și reciproc
- Virgulă mobilă: un număr se reprezintă printr-o pereche de reprezentări în virgulă fixă
  - permițând reprezentarea simultană de numere cu magnitudini și precizii diferite



# Notăția științifică; normalizare

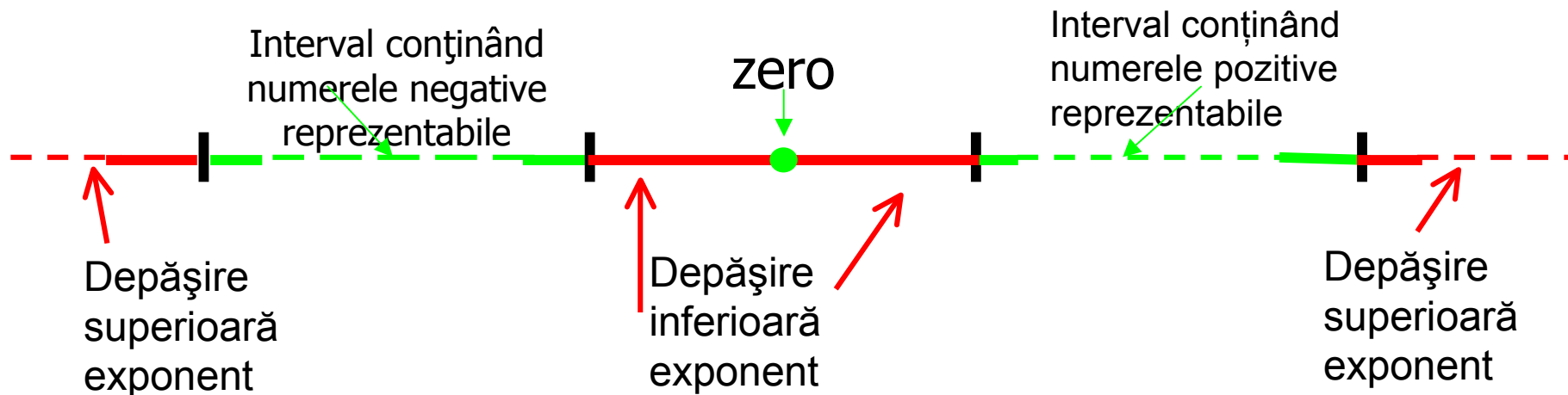
- Exemple
  - $7.015791043_{zece} \dots$ 
    - » scriere pozițională
  - $55312.1784_{zece} \times 10^{-105}$ 
    - » ar necesita 110 cifre în scriere pozițională, doar 14 aici.
  - $0.00000083_{zece} \times 10^4$ 
    - » Se poate scrie și eliminând zerourile semnificative
  - Numeroase scrieri posibile pentru același număr
- Notăția științifică:
  - o singură cifră la stânga virgulei
  - $0.031056_{zece} \times 10^7$
  - încă numeroase scrieri posibile pentru același număr
    - »  $m_1 \times B^{e_1} = m_2 \times B^{e_1+k}$ , unde  $m_2 = m_1 / B^{-k}$ , pt. oricare  $k \in \mathbf{Z}$ .
- Scriere normalizată: reprezentare în notație științifică *fără prefix de zerouri semnificative*
  - $6.15_{zece} \times 10^{-75}$ 
    - » pozițional  $\rightarrow$  78 de cifre semnificative, dintre care 75 zerouri
  - unică pentru un număr dat

# Notăția științifică în binar

- $1.101_{\text{doi}} \times 2^2$ 
  - $6,5_{\text{zece}}$
- $1.xxx\dots x_{\text{doi}} \times 2^{yyy}$ 
  - în binar, doar numărul 0 nu conține nici o cifră 1
    - definire specială a reprezentării lui 0
  - pentru oricare alt număr reprezentabil, cel mai semnificativ 1 trebuie să devină singura cifră de la partea întreagă
    - ceea ce, în general, alterează exponentul lui 2

# Reprezentări în virgulă mobilă

- Semnul (S): 0 sau 1
  - 1 bit
- Partea fracționară (f); mantisa este  $1+f$ 
  - mantisa are  $1+23$  (sau  $1+52$ ) biți
- Caracteristica (C)
  - $k=8$  sau  $k=11$  biți
  - $C = \text{exponent} + \text{exces}$
- $x = ( \pm ) 1.f \times B^{C - \text{exces}}$
- Primul 1 și baza 2 se subînțeleg/nu se reprezintă
  - » O figură similară pentru numerele reprezentabile în virgulă fixă ar indica un singur interval conținând numere reprezentabile, simetric față de 0 și cu cele două capete aflate fiecare într-unul din intervalele figurate aici cu verde.

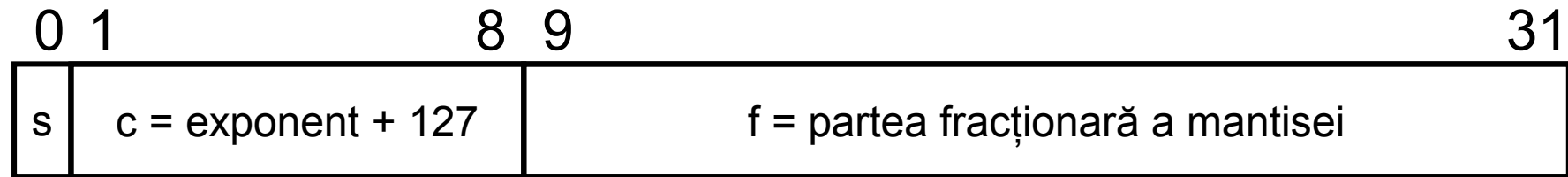


# Standardizare

- Esențială pentru portabilitate și pentru atașarea unei semantici generale a reprezentării în virgulă mobilă.
- Proces început în 1977, încheiat parțial în 1985.
- W. Kahan (University of Toronto).
- Prima implementare comercială a standardului IEEE (pe atunci, în curs de elaborare): 1981 - Intel 8087.

# Standardul IEEE 754 / 1985

- Precizie simplă: *float* în C/C++ (32 biți)



Margini în baza 10:  $1.2 \times 10^{-38} \rightarrow 3.4 \times 10^{38}$

Interesează nu doar structura reprezentării, ci mai ales *operațiile* ce se pot efectua cu reprezentări.

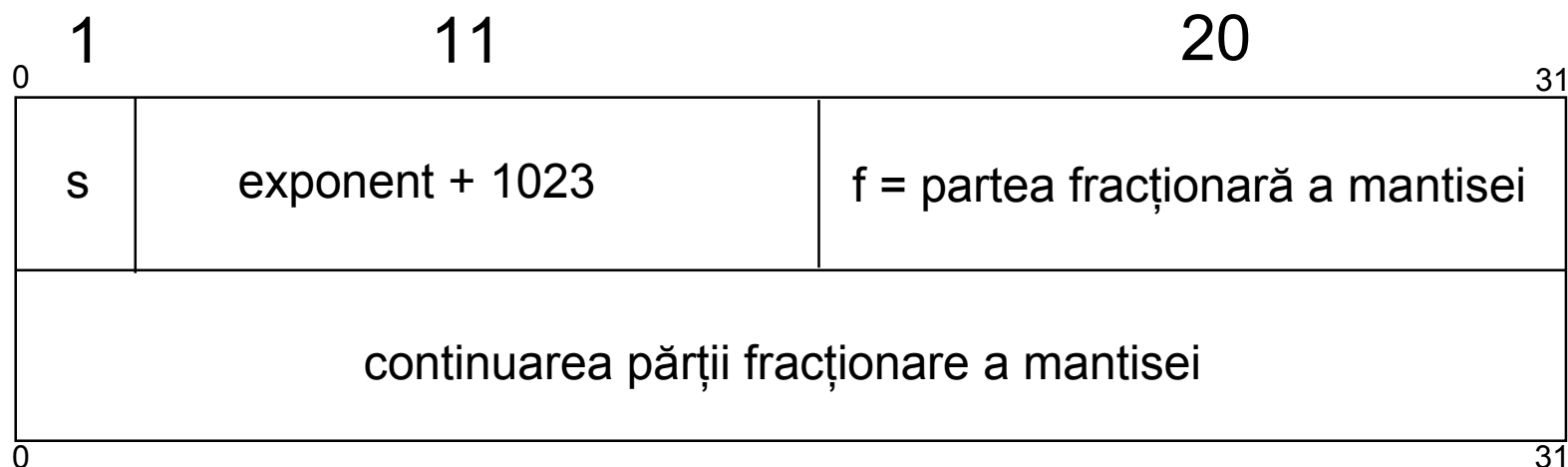
Exemple:

Compararea: reprezentarea în XS are avantajul inducerii ordinii naturale pe mulțimea reprezentării exponenților

Înmulțirea: adunarea exponenților la înmulțire – cu scăderea excesului

# Precizie dublă

- Precizie dublă: *double* în C/C++ (64 biți)



Margini în baza 10:  $1.7 \times 10^{-308} \rightarrow 1.7 \times 10^{308}$

Față de precizia simplă:

- Crește intervalul pentru exponent
- La exponent egal, crește acuratețea (precizia) reprezentării datorită lungimii mai mari a părții fracționare a mantisei

Două reprezentări în virgulă fixă =  
o reprezentare în virgulă mobilă

- Mantisa: reprezentare *modul și semn* (A+S) a coeficientului puterii bazei
- Caracteristica: reprezentare *în exces* a exponentului

- Există deci o valoare minimă  $e_{\min}$  și una maximă  $e_{\max}$  pentru exponent
  - de unde structura mulțimii numerelor reprezentabile
- Ordinea din reprezentare (S C f) facilitează compararea reprezentărilor
  - ordinea numerelor reprezentate coincide astfel cu ordinea lexicografică a reprezentărilor



# Reprezentări în norma IEEE 754

	Precizie simplă	Precizie dublă
Biți "precizie"	24	53
Exponent maxim	128 (pentru numere: 127)	1024 (pentru numere: 1023)
Exponent minim	-127 (pentru numere normalizate: -126)	-1023 (pentru numere normalizate: -1022)
Exces (exponent)	127	1023

# Un exemplu

- Cum se reprezintă numărul -7 în virgulă mobilă simplă precizie (IEEE 754) ?

1. **Semnul:** minus, deci 1

2. **Trecere în baza 2:**  $7_{zece} = 111_{doi}$

3. **Normalizare:**  $111_{doi} = 1.11_{doi} \times 2^2_{zece}$

4. **Calculul caracteristicii (pe 8 biți) :**  $(2 + 127)_{zece} = 129_{zece} = 10000001_{doi}$

5. **Reprezentarea:**

1 100.0000.1 110.0000.0000.0000.0000.0000<sub>doi</sub>

C0E00000<sub>hexazecimal</sub>

## Încă un exemplu

- Cum se scrie pozițional în baza zece numărul reprezentat în simplă precizie IEEE 754 prin  $C1F00000_{\text{hexazecimal}}$  ?
- Scriere în binar:**  $C1F00000_{\text{hexazecimal}} = 1100\ 0001\ 1111\ 0000\ 0000\ 0000\ 0000\ 0000_{\text{doi}}$
  - Semn:**  $1$ , deci  $-$  (număr negativ)
  - Caracteristica:**  $10000011_{\text{doi}} = 131_{\text{zece}}$
  - Exponentul:**  $131 - 127 = 4$
  - Mantisa:**  $(1 + 0,111)_{\text{doi}} = 1,111_{\text{doi}}$
  - Valoarea:**  $-1,111 \times 2^4 = -11110_{\text{doi}} = -30_{\text{zece}}$

# Aritmetica extinsă (principii)

- Aritmetica reală uzuală
  - proiectată pe mulțimea numerelor reale reprezentabile
  - cu operațiile uzuale
- La care se adaugă:
  - reprezentare pentru  $\infty$  și reguli elementare de calcul cu acesta ( $a / \infty$ ,  $\infty + \infty$ )
  - reprezentări pentru rezultatul operațiilor nedefinite (NaN) și reguli de propagare a acestuia (NaN *op* x = NaN)

# Aritmetica extinsă - exemplu

- Calculul lui arccos cu formula:  
$$\arccos(x) = 2 \arctan(\sqrt{(1-x)/(1+x)})$$
- $\arccos(-1) = ?$
- $1+x \rightarrow 0 \Rightarrow 2/(1+x) \rightarrow \infty \Rightarrow$   
 $\arctan((1-x)/(1+x)) \rightarrow \pi/2$
- Aceste relații fac parte din aritmetica în virgulă mobilă IEEE 754
- Rezultat - corect:  $\arccos(-1) = \pi$

# Tipuri de valori în virgulă mobilă

- În fiecare caz de mai jos, S este +1 sau -1 după cum bitul semn este 0 sau 1

Tip valoare	e	f	Valoare
normalizată	$e_{\min} < e < e_{\max}$	f oarecare	$(-1)^s \times 1.f \times 2^e$
denormalizată	$e = e_{\min}$	$f \neq 0$	$(-1)^s \times 0.f \times 2^e$
zero	$e = 0$	$f = 0$	S 0
infinit	$e = e_{\max}$	$f = 0$	S $\infty$
NaN	$e = e_{\max}$	$f \neq 0$	NaN

# Depășiri

- **Depășire inferioară:** în forma normalizată a numărului, exponentul **negativ** nu poate fi reprezentat în câmpul caracteristicii
  - numărul va fi considerat 0
- **Depășire superioară:** în forma normalizată a numărului, exponentul **pozitiv** este prea mare pentru a putea fi reprezentat în câmpul caracteristicii
  - numărul va fi considerat  $\pm\infty$

# Reprezentări denormalizate

- număr mai mic în modul decât cea mai mică reprezentare normalizată
  - se renunță la normalizare
  - mantisa va fi 0.f, în loc de 1.f
  - iar exponentul va avea valoarea minimă
    - 127 pentru simplă precizie
    - 1023 pentru dublă precizie
  - astfel se pot reprezenta numere mai mici



# Aritmetica în virgulă mobilă

- În general: fie  $x = x_m \times 2^{x_e}$  și  $y = y_m \times 2^{y_e}$ 
  - Relațiile de mai jos se referă la notația științifică, nu la reprezentarea în standard IEEE
    - »  $x_m$  și  $y_m$  sunt mantisele, iar  $x_e$  și  $y_e$  exponenții (nu caracteristicile)
  - $x + y = (x_m \times 2^{x_e - y_e} + y_m) \times 2^{y_e}$ , dacă  $x_e \leq y_e$
  - $x - y = (x_m \times 2^{x_e - y_e} - y_m) \times 2^{y_e}$ , dacă  $x_e \leq y_e$
  - $x \times y = (x_m \times y_m) \times 2^{x_e + y_e}$
  - $x : y = (x_m : y_m) \times 2^{x_e - y_e}$
- Operațiile sunt însă mai complicate decât o arată formulele

# Adunarea în virgulă mobilă

1. Se *compară exponenții* termenilor adunării
  - dacă aceștia sunt diferiți, atunci partea fracționară a numărului mai mic se deplasează spre dreapta (denormalizare!) până când exponentul său devine egal cu exponentul mai mare
  - evident, unitatea hardware care face calculele operează în interior cu mai mulți biți decât reprezentarea standard
2. Se adună mantisele
  - aici se decide și **semnul** sumei
  - adunarea mantiselor se efectuează în complement față de 2
    - deci pentru numere negative, se face complementarea corespunzătoare, iar rezultatul – dacă este negativ – se complementează pentru a reveni la  $A+S$
3. Dacă este nevoie, se normalizează suma
  - fie decalând rezultatul spre dreapta și incrementând exponentul, fie decalând spre stânga și decrementând exponentul
4. Dacă se produce depășire → excepție → stop
5. Rotunjește mantisa la numărul permis de biți
  - dacă astfel s-a produs denormalizare, reia de la 3

# Temă

- Să se urmărească pașii adunării în virgulă mobilă pentru reprezentările numerelor scrise în baza zece ca 0,75 și -0,375.

Se va considera că semnul, exponentul și mantisa sunt obținute ca pentru IEEE 754, dar se reprezintă pe 1 bit, 8 biți, respectiv 4 biți.

# Înmulțirea în virgulă mobilă

1. Se calculează exponentul rezultatului adunând exponenții celor doi factori
  - se adună caracteristicile și se scade excesul
2. Se înmulțesc mantisele
3. Se normalizează rezultatul
  - dacă se produce depășire → excepție → stop
4. Se fac rotunjirile necesare
  - dacă se produce denormalizare, reia de la 3
5. Se determină semnul rezultatului

# Temă

- Să se urmărească pașii înmulțirii în virgulă mobilă pentru reprezentările numerelor scrise în baza zece ca 0,75 și 0,375.

Reprezentările sunt cele din standardul IEEE 754.

## Capitolul al V-lea

# ARHITECTURA ȘI ORGANIZAREA CALCULATORULUI

V.1.

CALCULATORARE VON NEUMANN

# Calculatoare von Neumann

- program memorat
  - memorie infinită (ideal), timp de acces egal
    - realizată practic prin *ierarhii de memorie*
- la execuție, după o instrucțiune **i** urmează
  - instrucțiunea memorată imediat după ea (regula)
  - eventual instrucțiunea indicată de **i** (dacă **i** este instrucțiune **de control**)
- adresa instrucțiunii următoare se află într-un registru – PC
- în fiecare moment, o singură instrucțiune este încărcată pentru execuție



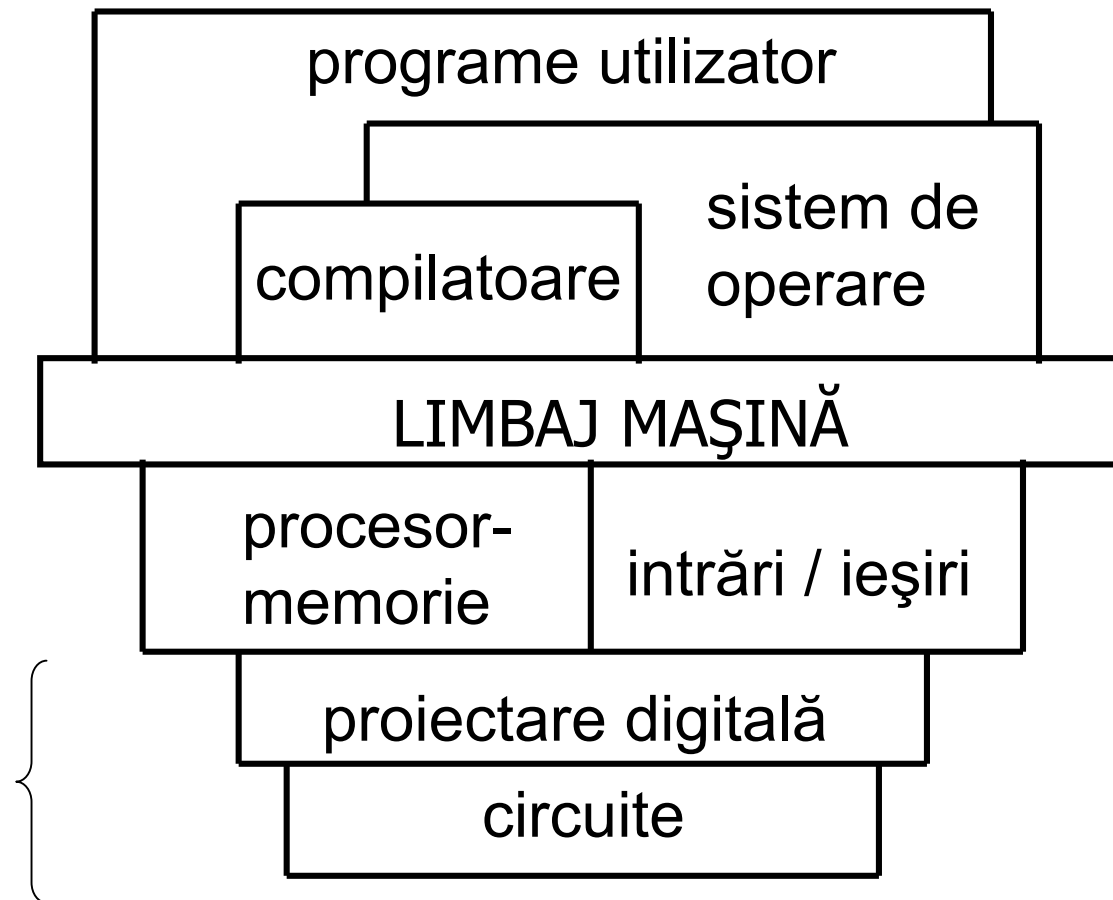
# Calculatoare von Neumann

- Anterior:
  - conceptul de automatizare a operațiilor luate separat (Pascal, Leibniz)
  - conceptul de program – exterior (Babbage)
  - conceptul de calcul ramificat / control
- Conceptul de program memorat
  - John von Neumann et al (1946)
  - concept arhitectural fundamental - procesoarele moderne

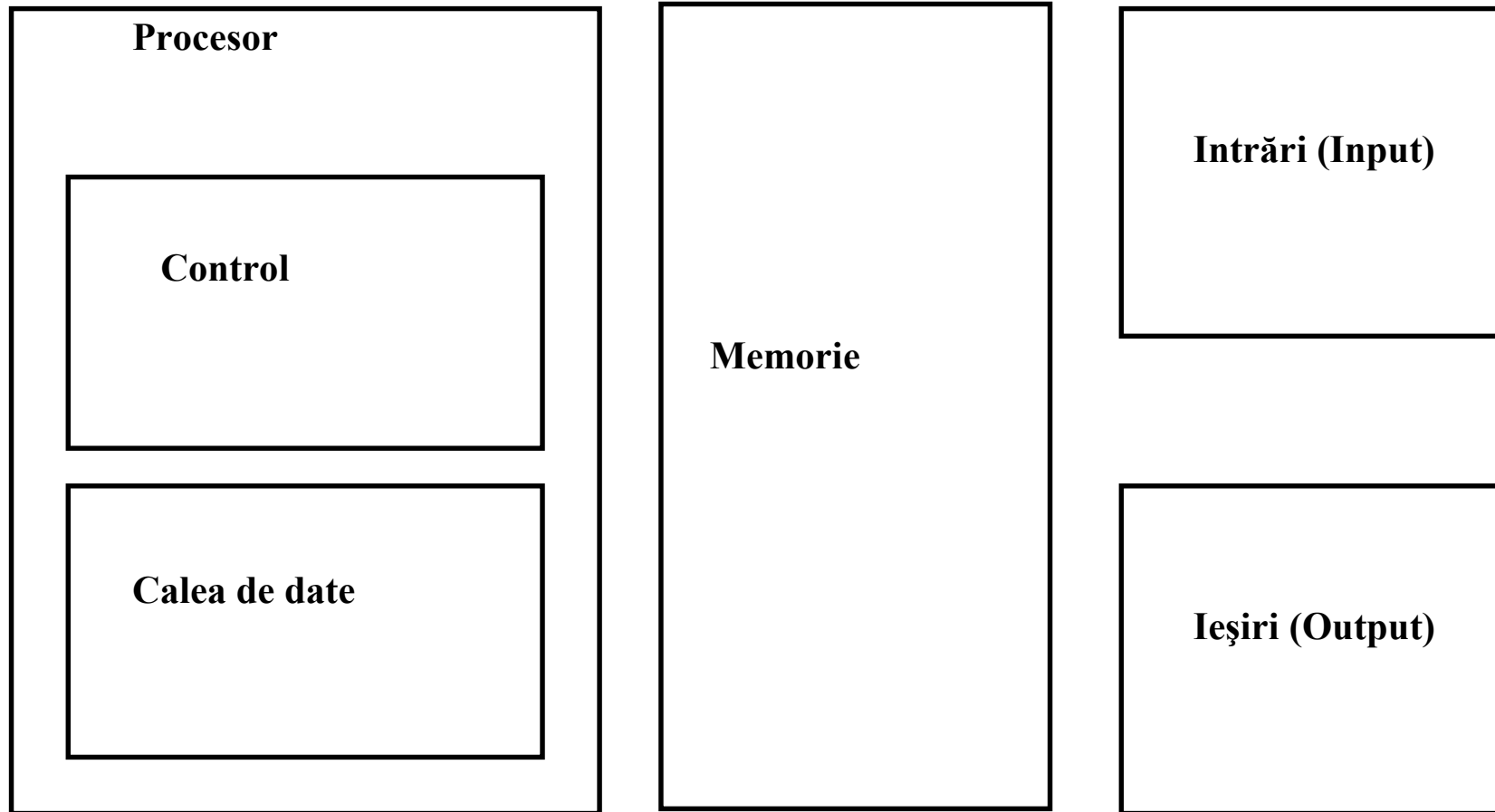
# Calculatoare von Neumann

- Programul și datele - stocate în (aceeași) memorie
  - ideal: infinită, omogenă (locații la fel de rapid accesabile)
  - practic: o ierarhie de memorii, fiecare fiind omogenă
- Program counter (PC) indică locul din memorie al instrucțiunii de executat
  - conținutul PC este actualizat la execuția fiecărei instrucțiuni
  - o dată sau de două ori
- Instrucțiunile programului sunt aduse pe rând din locații de memorie în procesor
  - regula: locații succesive → incrementare PC
  - excepția: instrucțiuni de salt
  - ordine fizică și ordine logică

# Arhitectura unui sistem de calcul



# Componentele hardware ale unui calculator



# Organizarea unui calculator

