

Reprezentarea în complement față de 1

- notație: C_1

$$\begin{aligned} \text{val}_{C_1}^{n,m}(a_{n-1}a_{n-2}\dots a_1a_0a_{-1}\dots a_{-m}) &= \\ &= \begin{cases} a_{n-2}\times 2^{n-2} + \dots + a_{-m}\times 2^{-m} & \text{dacă } a_{n-1}=0 \\ (a_{n-2}\times 2^{n-2} + \dots + a_{-m}\times 2^{-m}) - (2^{n-1} - 2^{-m}) & \text{dacă } a_{n-1}=1 \end{cases} \end{aligned}$$

- temă: demonstrați că valoarea este negativă pentru $a_{n-1} = 1$
 - deci a_{n-1} reprezintă semnul

Complement față de 1 - limite

- pe $n+m$ biți sunt 2^{n+m} reprezentări diferite
 - dar numai $2^{n+m} - 1$ numere diferite
 - redundantă: $\text{val}_{C_1}^{n,m}(00\dots 0) = \text{val}_{C_1}^{n,m}(11\dots 1) = 0$

- valorile extreme reprezentabile

$$\max_{C_1}^{n,m} = \text{val}_{C_1}^{n,m}(01\dots 1) = 2^{n-1} - 2^{-m}$$

$$\min_{C_1}^{n,m} = \text{val}_{C_1}^{n,m}(10\dots 0) = -(2^{n-1} - 2^{-m})$$

- deci numerele reprezentabile sunt în intervalul $[-(2^{n-1} - 2^{-m}); +(2^{n-1} - 2^{-m})]$

Complement față de 1 - precizie

- numerele reprezentabile exact încep cu $\min = -(2^{n-1} - 2^{-m})$
 - și continuă cu pasul 2^{-m}
- celelalte numere din interval - aproximare
 - eroarea - cel mult 2^{-m}
- deci precizia reprezentării este 2^{-m}
- pentru $n+m$ fixat
 - numere mai mari = precizie mai slabă și invers

Complementare

- reprezentările pentru numerele pozitive - ușor de determinat
- pentru numere negative - mai greu
- există o relație între reprezentarea numărului q și cea a numărului $-q$?
- da: reprezentarea lui $-q$ se obține negând toți biții din reprezentarea lui q
 - operație comutativă - valabilă și pentru $q < 0$

Example (1)

$$\text{val}_{C_1}^{8,0}(00110011) = 2^5 + 2^4 + 2^1 + 2^0 = 51$$

$$\text{val}_{C_1}^{6,2}(00110011) = 2^3 + 2^2 + 2^{-1} + 2^{-2} = 12,75$$

sau

$$\text{val}_{C_1}^{6,2}(00110011) = \text{val}_{C_1}^{8,0}(00110011) : 2^2 = 51 : 4 = 12,75$$

$$\text{val}_{C_1}^{4,4}(00110011) = 2^1 + 2^0 + 2^{-3} + 2^{-4} = 3,1875$$

sau

$$\text{val}_{C_1}^{4,4}(00110011) = \text{val}_{C_1}^{8,0}(00110011) : 2^4 = 51 : 16 = 3,1875$$

Example (2)

$$\text{val}_{C_1}^{8,0}(11001100) = (2^6 + 2^3 + 2^2) - (2^7 - 2^0) = -51$$

$$\text{val}_{C_1}^{4,4}(11001100) = (2^2 + 2^{-1} + 2^{-2}) - (2^3 - 2^{-4}) = -3,1875$$

sau

$$\text{val}_{C_1}^{4,4}(11001100) = \text{val}_{C_1}^{8,0}(11001100) : 2^4 = -51 : 16 = -3,1875$$

$$\text{min}_{C_1}^{8,0} = \text{val}_{C_1}^{8,0}(10000000) = 0 - (2^7 - 2^0) = -127$$

$$\text{min}_{C_1}^{4,4} = \text{val}_{C_1}^{4,4}(10000000) = 0 - (2^3 - 2^{-4}) = -7,9375$$

sau

$$\text{min}_{C_1}^{4,4} = \text{min}_{C_1}^{8,0} : 2^4 = -127 : 16 = -7,9375$$

Example (3)

$$\max_{C_1}^{8,0} = \text{val}_{C_1}^{8,0}(01111111) = 127$$

$$\max_{C_1}^{4,4} = \text{val}_{C_1}^{4,4}(01111111) = 7,9375$$

sau

$$\max_{C_1}^{4,4} = \max_{C_1}^{8,0} : 2^4 = 127 : 16 = 7,9375$$

- intervale reprezentabile
 - $C_1^{8,0}$: $[-127; 127] \rightarrow 255$ numere, din 1 în 1
 - $C_1^{4,4}$: $[-7,9375; 7,9375] \rightarrow 255$ numere, din 0,0625 în 0,0625 ($=1:16$)

Operații în C_1

- putem aduna două numere în C_1 cu ajutorul unui sumator "clasic"?
- da, dar în doi pași
 - în al doilea pas, la rezultat se adună transportul obținut la primul pas
 - deci trebuie două sumatoare pentru o adunare
- scădere: adunăm descăzutul cu simetricul scăzătorului

Reprezentarea în complement față de 2

- cerințe
 - reprezentare neredundantă
 - o singură reprezentare pentru 0
 - adunarea a două numere se poate realiza cu un singur sumator
 - la fel ca la numere fără semn
 - beneficiu - o singură operație de adunare implementată în procesor pentru tipuri de date cu semn și fără semn

Complement față de 2

- notăție: C_2

$$\begin{aligned} \text{val}_{C_2}^{n,m}(a_{n-1}a_{n-2}\dots a_1a_0a_{-1}\dots a_{-m}) &= \\ &= \begin{cases} a_{n-2}\times 2^{n-2} + \dots + a_{-m}\times 2^{-m} & \text{dacă } a_{n-1}=0 \\ (a_{n-2}\times 2^{n-2} + \dots + a_{-m}\times 2^{-m}) - 2^{n-1} & \text{dacă } a_{n-1}=1 \end{cases} \end{aligned}$$

- temă: demonstrați că valoarea este negativă pentru $a_{n-1} = 1$
 - deci a_{n-1} reprezintă semnul

Complement față de 2 - limite

- pe $n+m$ biți sunt 2^{n+m} reprezentări diferite
 - și 2^{n+m} numere diferite
 - $00\dots0$ - singura reprezentare pentru 0

- valorile extreme reprezentabile

$$\max_{C_2}^{n,m} = \text{val}_{C_2}^{n,m}(01\dots1) = 2^{n-1} - 2^{-m}$$

$$\min_{C_2}^{n,m} = \text{val}_{C_2}^{n,m}(10\dots0) = -2^{n-1}$$

- deci numerele reprezentabile sunt în intervalul $[-2^{n-1}; +(2^{n-1} - 2^{-m})]$ - asimetric

Complement față de 2 - precizie

- numerele reprezentabile exact încep cu $\min = -2^{n-1}$
 - și continuă cu pasul 2^{-m}
- celelalte numere din interval - aproximare
 - eroarea - cel mult 2^{-m}
- deci precizia reprezentării este 2^{-m}
- pentru $n+m$ fixat
 - numere mai mari = precizie mai slabă și invers

Complementare (1)

- există o relație între reprezentarea numărului q și cea a numărului $-q$?
- da: reprezentarea lui $-q$ este complementul față de 2 al reprezentării lui q
 - se neagă toți biții și se adună 0...01
 - la fel ca la C_1 , operația este comutativă - se poate aplica indiferent de semnul q

Complementare (2)

- exemplu

$q = 77$ are reprezentarea 01001101 în $C_2^{8,0}$

$-q = -77$ are reprezentarea $10110010 + 00000001$
 $= 10110011$

- temă

– reprezentarea în C_2 pe N biți a numărului întreg negativ q este de fapt reprezentarea pe N biți a numărului $q + 2^N = 2^N - |q|$

Example (1)

$$\text{val}_{C_2}^{8,0}(00110011) = 2^5 + 2^4 + 2^1 + 2^0 = 51$$

$$\text{val}_{C_2}^{6,2}(00110011) = 2^3 + 2^2 + 2^{-1} + 2^{-2} = 12,75$$

sau

$$\text{val}_{C_2}^{6,2}(00110011) = \text{val}_{C_2}^{8,0}(00110011) : 2^2 = 51 : 4 = 12,75$$

$$\text{val}_{C_2}^{4,4}(00110011) = 2^1 + 2^0 + 2^{-3} + 2^{-4} = 3,1875$$

sau

$$\text{val}_{C_2}^{4,4}(00110011) = \text{val}_{C_2}^{8,0}(00110011) : 2^4 = 51 : 16 = 3,1875$$

Example (2)

$$\text{val}_{C_2}^{8,0}(11001101) = (2^6 + 2^3 + 2^2) - (2^7 - 2^0) = -51$$

$$\text{val}_{C_2}^{4,4}(11001101) = (2^2 + 2^{-1} + 2^{-2}) - (2^3 - 2^{-4}) = -3,1875$$

sau

$$\text{val}_{C_2}^{4,4}(11001101) = \text{val}_{C_2}^{8,0}(11001101) : 2^4 = -51 : 16 = -3,1875$$

$$\text{min}_{C_2}^{8,0} = \text{val}_{C_2}^{8,0}(10000000) = 0 - 2^7 = -128$$

$$\text{min}_{C_2}^{4,4} = \text{val}_{C_2}^{4,4}(10000000) = 0 - 2^3 = -8$$

sau

$$\text{min}_{C_2}^{4,4} = \text{min}_{C_2}^{8,0} : 2^4 = -128 : 16 = -8$$

Example (3)

$$\max_{C_2}^{8,0} = \text{val}_{C_2}^{8,0}(01111111) = 127$$

$$\max_{C_2}^{4,4} = \text{val}_{C_2}^{4,4}(01111111) = 7,9375$$

sau

$$\max_{C_2}^{4,4} = \max_{C_2}^{8,0} : 2^4 = 127 : 16 = 7,9375$$

- intervale reprezentabile
 - $C_2^{8,0}$: $[-128; 127] \rightarrow 256$ numere, din 1 în 1
 - $C_2^{4,4}$: $[-8; 7,9375] \rightarrow 256$ numere, din 0,0625 în 0,0625 ($=1:16$)

Concluzii

- C_2 este reprezentarea utilizată cel mai des
 - neredundantă
 - adunarea/scăderea - implementate la fel ca la numere fără semn
- în practică - tipuri de date întregi din limbajele de programare
 - caz particular ($m=0$)
 - pentru numere reale se utilizează reprezentările în virgulă mobilă

IV.5. Depășiri pentru operații cu reprezentări în virgulă fixă

Depășiri

- nu sunt suficienți biți la partea întreagă pentru numărul de reprezentat
 - numărul este în afara intervalului reprezentabil
- problemă
 - având două numere reprezentabile, rezultatul unei operații efectuate asupra lor poate să nu fie reprezentabil - *depășire*
 - când se întâmplă și cum detectăm așa ceva?

Trecerea la reprezentări mai lungi

- având un număr reprezentat pe n biți, cum obținem reprezentarea sa pe $n+k$ biți?
 - adăugare de cifre ne semnificative la partea întreagă; partea fracționară nu e afectată
- $A+S$: se adaugă k zerouri imediat la dreapta cifrei semn
- C_1, C_2 : se repetă cifra semn de k ori imediat la dreapta cifrei semn

Example

	număr	
codificare	51	-51
$A+S^{8,0}$	00110011	10110011
$A+S^{16,0}$	0000000000110011	1000000000110011
$C_1^{8,0}$	00110011	11001100
$C_1^{16,0}$	0000000000110011	1111111111001100
$C_2^{8,0}$	00110011	11001101
$C_2^{16,0}$	0000000000110011	1111111111001101

Trecerea la reprezentări mai scurte

- folosim aceste rezultate pentru a răspunde la problema inversă
- având un număr reprezentat pe n biți, poate fi reprezentat pe $n-k$ biți?
 - da, dacă și numai dacă cei k biți de la dreapta celui de semn au valorile ca mai înainte
 - în acest caz, ei pot fi eliminați din reprezentare

Operații în C_2

- în continuare vom discuta doar cazul C_2
 - reprezentarea cel mai des folosită
- restricții impuse de calculator asupra operațiilor cu reprezentări
 - termenii sumei și rezultatul se reprezintă pe același număr de biți
 - termenii înmulțirii se reprezintă pe același număr de biți, iar rezultatul pe număr dublu de biți

Definiția depășirii

- fie o reprezentare dată și op o operație cu numere
- pe $n+m$ biți, numerele reprezentabile sunt în intervalul $[\min; \max]$
- fie două numere $a, b \in [\min; \max]$
- operația op aplicată numerelor a și b produce depășire dacă

$$a \ op \ b \notin [\min; \max]$$

Example (1)

- în continuare vom folosi reprezentarea C_2 cu $n=4$, $m=0$

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

- bitul "suplimentar" este ignorat (doar 4 biți)
- de fapt este bitul de transport

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

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

- rezultatul este corect - nu avem depășire

Example (2)

$$0111 + 0111 = 1110 \rightarrow 1110$$

– nu avem bit "suplimentar" (transportul este 0)

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

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

– rezultat incorect - se produce depășire

- concluzie - bitul de transport nu oferă informații privind depășirea
 - trebuie căutată altă formă de detecție

Condiția de depășire

- nu putem folosi direct definiția depășirii
 - numerele nu sunt disponibile
- constatare
 - depășire se poate produce la adunare doar când ambii operanzi au același semn
 - iar reprezentarea rezultatului indică semn opus
- temă: nu se poate produce depășire la adunarea a două numere de semn opus

Suma algebrică în C_2 (1)

- Teorema 1

dacă numerele a și b sunt reprezentabile în $C_2^{n,m}$, atunci $a \pm b$ sunt reprezentabile în $C_2^{n+1,m}$

- Lemă

dacă $a = \text{val}_{C_2}^{n+1,m}(\alpha_n \alpha_{n-1} \dots \alpha_1 \alpha_0 \alpha_{-1} \dots \alpha_{-m})$ și $\alpha_n = \alpha_{n-1}$
atunci $a = \text{val}_{C_2}^{n,m}(\alpha_{n-1} \dots \alpha_1 \alpha_0 \alpha_{-1} \dots \alpha_{-m})$

Suma algebrică în C_2 (2)

- fie reprezentările

$$\alpha = \alpha_{n-1}\alpha_{n-2}\dots\alpha_1\alpha_0\alpha_{-1}\dots\alpha_{-m}$$

$$\beta = \beta_{n-1}\beta_{n-2}\dots\beta_1\beta_0\beta_{-1}\dots\beta_{-m}$$

- definim suma lor formală $\gamma = \alpha + \beta$ ca

$$\gamma = \gamma_n\gamma_{n-1}\gamma_{n-2}\dots\gamma_1\gamma_0\gamma_{-1}\dots\gamma_{-m}$$

adică

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

Suma algebrică în C_2 (3)

- Teorema 2

dacă suma algebrică a numerelor reprezentate de α și β nu produce depășire, atunci reprezentarea rezultatului este

$$\gamma_{n-1}\gamma_{n-2}\cdots\gamma_1\gamma_0\gamma_{-1}\cdots\gamma_{-m}$$

- Teorema 3

suma algebrică a numerelor reprezentate de α și β nu produce depășire dacă cifrele transport C_{n-1} și C_n ale rezultatului coincid

Consecințe

- adunarea a două numere în C_2 se poate realiza utilizând un sumator "clasic"
 - biții de semn se adună la fel ca toți ceilalți biți
- testarea depășirii în C_2 se poate realiza adăugând la sumator o poartă NXOR
 - în care intră biții transport C_{n-1} și C_n
 - nu este deci necesar să fie cunoscute numerele

IV.4. Reprezentări în virgulă mobilă

Probleme cu reprezentările în virgulă fixă

- lungimea totală $n+m$ este fixată prin hardware
- dar, în virgulă fixă, atât n cât și m sunt la rîndul lor fixate
 - deci magnitudinea și precizia sunt prestabilite și nu pot fi modificate
 - dacă dorim o precizie mai bună și suntem dispuși să reducem magnitudinea (sau invers)?

Notăția științifică

- [illegible]

Notăția științifică în binar

- cifra semnificativă dinaintea virgulei poate fi doar 1
 - deci în practică nu este necesară memorarea sa
- excepție - reprezentarea numărului 0
 - doar cifre de 0
- scriere normalizată (număr nenul)
 $1.xx...x \times 2^y$
 - baza 2 este implicită - nu trebuie memorată

Reprezentări în virgulă mobilă

- componente
 - semnul (S): 0 sau 1 (+ sau -)
 - mantisa (M): $1.xx...x$
 - de obicei se folosește partea fracționară (f)
 $M = 1 + f; f = 0.xx...x$
 - caracteristica
 - reprezentarea exponentului în exces

$$N = (-1)^S \times 1.f \times 2^C - \text{exces}$$

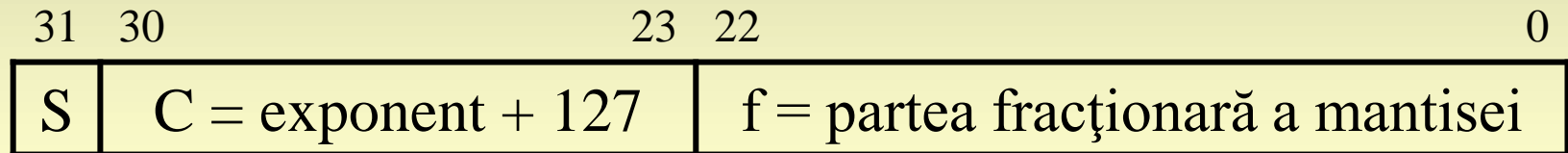
Limite

- numărul de biți al caracteristicii este fixat
 - deci există o valoare minimă și una maximă pentru exponent
- depășire superioară - exponent prea mare
 - numărul este considerat $\pm\infty$
- depășire inferioară - exponent prea mic
 - numărul este considerat 0
- tipul depășirii nu depinde de semn

Standardizare

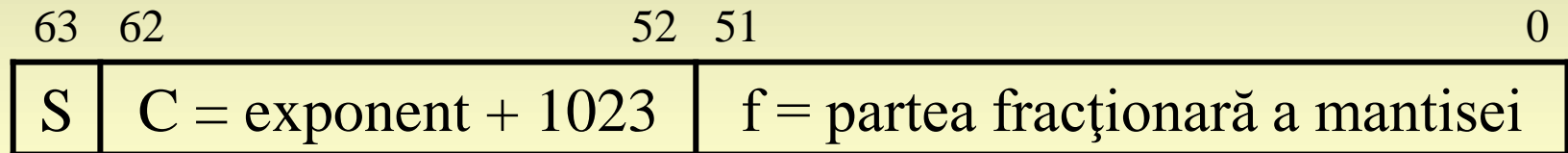
- esențială pentru portabilitate
- standardul IEEE 754/1985
 - elaborat între 1977 și 1985
 - prima implementare comercială: Intel 8087
- 2 variante principale
 - simplă precizie (32 biți)
 - dublă precizie (64 biți)
 - au fost proiectate și unele extensii

Simplă precizie



- corespunde tipului *float* din C/C++
- limite în baza 10
 - minim: $\approx 1.2 \times 10^{-38}$
 - orice număr mai mic în modul va fi considerat 0
 - maxim: $\approx 3.4 \times 10^{38}$
 - orice număr mai mare în modul va fi considerat $\pm\infty$

Dublă precizie



- corespunde tipului *double* din C/C++
- limite în baza 10
 - minim: $\approx 1.7 \times 10^{-308}$
 - maxim: $\approx 1.7 \times 10^{308}$
- magnitudine mai mare
- precizie superioară

Structură

- de fapt, reprezentarea în virgulă mobilă este formată din două reprezentări în virgulă fixă
 - semnul și mantisa - reprezentare modul-semn
 - caracteristica - reprezentare în exces
- de ce câmpurile sunt în ordinea S,C,f?
 - pentru a compara două reprezentări, câmpurile trebuie luate în considerare în această ordine

Caracteristici IEEE 754/1985

	Simplă precizie	Dublă precizie
Biți semn+mantisă	24	53
Exponent maxim	128	1024
• numere finite	127	1023
Exponent minim	-127	-1023
• numere normalizate	-126	-1022
Exces caracteristică	127	1023

Exemplu 1

- fie numărul -23.25
 - cum se reprezintă în simplă precizie?
- semnul: **1** (negativ)
- scriere în baza 2: $-23.25_{(10)} = -10111.01_{(2)}$
- normalizare: $10111.01 = 1.011101 \times 2^4$
- caracteristica: $4 + 127 = 131 = 10000011_{(2)}$
- reprezentarea
 $\mathbf{1} \mathbf{10000011} \mathbf{0111010...0}_{(2)} = \mathbf{C1BA0000}_{(16)}$

Exemplu 2

- ce număr corespunde reprezentării $42D80000_{(16)}$ (simplă precizie)?

$$42D80000_{(16)} = \textcolor{red}{0} \textcolor{blue}{10000101} \textcolor{green}{10110000...0}_{(2)}$$

$$S = \textcolor{red}{0} \rightarrow \text{număr pozitiv}$$

$$C = \textcolor{blue}{10000101}_{(2)} = 133_{(10)} \Rightarrow e = 133 - 127 = 6$$

$$M = 1 + 0.\textcolor{green}{1011} = 1.\textcolor{green}{1011}$$

- numărul: $+1.1011 \times 2^6 = 1101100_{(2)} = 108_{(10)}$

Aritmetica extinsă

- în plus față de aritmetica numerelor reale
 - reprezentarea numărului ∞ și definirea regulilor elementare de calcul cu acesta
 - $x / \infty, x \times \infty, \infty \pm \infty$
 - reprezentare pentru rezultatul operațiilor nedefinite (NaN - Not a Number) și definirea regulilor de propagare a acestuia
 - $\text{NaN } op \ x = \text{NaN}, \forall op$
- utilizare - bibliotecile de funcții matematice

Exemplu

- calculul funcției arccos cu formula

$$\arccos(x) = 2 \cdot \arctan \sqrt{(1-x)/(1+x)}$$

- care este valoarea $\arccos(-1)$?

$$x = -1 \Rightarrow (1-x)/(1+x) = 2/0 = \infty \Rightarrow$$

$$\Rightarrow \arctan \sqrt{(1-x)/(1+x)} = \pi/2$$

- răspuns: $\arccos(-1) = \pi$
 - nu ar fi fost posibil de obținut fără aritmetica extinsă

Tipuri de valori în virgulă mobilă

tip valoare	exponent (e)	f	valoare
normalizată	$e_{\min} < e < e_{\max}$	orice valoare	$(-1)^S \times 1.f \times 2^e$
denormalizată	$e = e_{\min}$	$f \neq 0$	$(-1)^S \times 0.f \times 2^e$
zero	$e = e_{\min}$	$f = 0$	$S \ 0 \ (= 0)$
infinit	$e = e_{\max}$	$f = 0$	$S \ \infty \ (\pm\infty)$
NaN	$e = e_{\max}$	$f \neq 0$	NaN