

Lucrarea de laborator nr. 2

I. Scopul lucrării

Reprezentarea numerelor reale în calculator. Erori de rotunjire.

II. Conținutul lucrării

1. Reprezentarea numerelor reale sub formă normalizată.
2. Reprezentarea numerelor reale în virgulă mobilă.
3. Erori care apar ca urmare a limitelor de reprezentare.

III. Prezentarea lucrării

III.1. Reprezentarea numerelor reale sub formă normalizată.

Forma normalizată a un număr real nenul x este o presupune următoarea reprezentare

$$x = m \times b^{er}, \text{ unde } \begin{cases} b = \text{baza} \\ m = \text{mantisa} \\ er = \text{exponentul} \end{cases}$$

cu $0,1_b \leq |m|_b < 1_b$ (ceea ce înseamnă că mantisa este un număr subunitar cu prima cifră după virgulă diferită de zero).

Pentru a scrie numărul sub formă normalizată trebuie găsite deci mantisa și exponentul. **Mantisa** se obține deplasând virgula în fața primei cifre nenule ce apare în scrierea numărului (în baza b). **Exponentul** se ia egal cu numărul de poziții cu care s-a deplasat virgula precedat de semnul + dacă deplasarea s-a făcut de la stânga la dreapta, și de semnul – dacă deplasarea s-a făcut de la stânga la dreapta. Astfel

- dacă x este reprezentat în baza b sub forma

$$x_b = \pm a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m}, \dots \text{ cu } a_n \neq 0,$$

atunci forma normalizată este

$$x_b = \pm 0, a_n a_{n-1} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-m} \dots \times b^{n+1}$$

- dacă x este reprezentat în baza b sub forma

$$x_b = \pm 0, a_{-1} a_{-2} \dots a_{-m}, \dots \text{ cu } a_{-1} \neq 0,$$

atunci forma normalizată este

$$x_b = \pm 0, a_1 a_2 \dots a_m \dots \times b^0$$

- dacă x este reprezentat în baza b sub forma

$x_b = \pm 0, a_1 \dots a_i a_{i-1} \dots a_m, \dots$ cu $a_1 = \dots = a_i = 0$ și $a_{i-1} \neq 0$, atunci forma normalizată este

$$x_b = \pm 0, a_{i-1} a_{i-2} \dots a_m \dots \times b^{-i}$$

III.2. Reprezentarea numerelor reale în virgulă mobilă.

Una dintre cele mai răspândite reprezentări internă (în PC-uri) a numerelor reale este **reprezentarea în virgulă mobilă**. Reprezentarea în virgulă mobilă presupune existența unei **baze b** (întotdeauna presupusă pară) și a unei **precizii p** . Un număr în virgulă mobilă este un număr de forma

$$(\alpha_0 + \frac{\alpha_1}{b} + \frac{\alpha_2}{b^2} + \dots + \frac{\alpha_{p-1}}{b^{p-1}}) b^E, \quad 0 \leq \alpha_k < b, \text{ pentru orice } k = \overline{0, p-1}, E \in \mathbf{Z}.$$

Mai precis, denumirea de număr în virgulă mobilă va fi utilizată pentru numerele reale care se reprezintă exact sub forma de mai sus. În această reprezentare $\alpha_0, \alpha_1, \dots, \alpha_{p-1}$ se numesc cifre semnificative. Fiecărei reprezentări în virgulă mobilă i se asociază două numere întregi, E_{\min} și E_{\max} , ce reprezintă **valorile limită** permise pentru exponentul E ($E_{\min} \leq E \leq E_{\max}$). Reprezentarea în virgulă mobilă se numește normalizată dacă se impune condiția ca cifra cea mai semnificativă α_0 să fie nenulă. Reprezentarea normalizată are următoarele avantaje:

- reprezentarea fiecărui număr este unică
- nu de pierde cifre pentru reprezentarea primele zerourilor de la dreapta virgulei
- în sistemele binare (corespunzătoare bazei $b=2$) prima cifră poate să nu mai fie stocată (deoarece este întotdeauna 1).

Restricția $\alpha_0 \neq 0$, face imposibilă reprezentarea lui zero. O reprezentare naturală a lui zero este $1, 0 \cdot b^{E_{\min}-1}$.

Numărul de numere în virgulă mobilă normalizată este

$$2(b-1)b^{p-1}(E_{\max} - E_{\min} + 1) + 1.$$

Cel mai mic număr pozitiv normalizat se notează UFL (underflow level) și este

$$\text{UFL} = b^{E_{\min}}.$$

Cel mai mare număr normalizat se notează OFL (overflow level) și este

$$\text{OFL} = (b-1 + \frac{b-1}{b} + \frac{b-1}{b^2} + \dots + \frac{b-1}{b^{p-1}}) b^{E_{\max}}$$

$$= b^{E_{\max} + 1} \left(1 - \frac{1}{b^p}\right).$$

Ca urmare nu toate numerele reale sunt reprezentabile exact. Numerele prea mari pentru a fi reprezentate corespund unei depășiri superioare de capacitate (*overflow*), iar numerele prea mici unei depășiri inferioare de capacitate (*underflow*). Pentru a fi reprezentat un număr real x este aproximat cu un număr în virgulă mobilă pe care convenim să-l notăm $fl(x)$. Aproximarea lui x prin $fl(x)$ poartă numele de *rotunjire*, iar eroarea introdusă de *eroare de rotunjire*. Există mai multe modalități pentru rotunjire:

- *trunchiere (rotunjire prin tăiere)*: se rețin primele p cifre din reprezentarea normalizată a lui $x = (\alpha_0 + \frac{\alpha_1}{b} + \frac{\alpha_2}{b^2} + \dots + \frac{\alpha_{p-1}}{b^{p-1}} + \dots)b^E$; deci

$$fl(x) = (\alpha_0 + \frac{\alpha_1}{b} + \frac{\alpha_2}{b^2} + \dots + \frac{\alpha_{p-1}}{b^{p-1}})b^E.$$

- *rotunjire la cel mai apropiat număr în virgulă mobilă (rotunjire la par)*: $fl(x)$ este cel mai apropiat număr în virgulă mobilă de x ; în caz de egalitate (dacă există două numere în virgulă mobilă egal depărtate de x) se consideră acel număr în virgulă mobilă a cărui ultimă cifră este pară. Rotunjirea la par determină o acuratețe mai mare a reprezentării. Acuratețea sistemului în virgulă mobilă este caracterizată de așa numita *precizie a mașinii* (sau *epsilon mașină*), notată ϵ_{mach} . Precizia a mașinii este definită ca cel mai mic număr pozitiv ϵ cu proprietatea că

$$fl(1 + \epsilon) > 1.$$

Dacă regula de rotunjire este trunchierea atunci

$$\epsilon_{mach} = b^{1-p},$$

iar dacă regula de rotunjire este rotunjirea la par atunci

$$\epsilon_{mach} = \frac{1}{2} b^{1-p}.$$

Eroarea relativă maximă cu care $fl(x)$ aproximează x este dată de

$$\left| \frac{fl(x) - x}{x} \right| \leq \epsilon_{mach}.$$

Deși amândouă sunt "mici", precizia mașinii (ϵ_{mach}) și cel mai mic număr pozitiv normalizat UFL (în reprezentare în virgulă mobilă fixată) nu trebuie confundate. De obicei $E_{\min} < -p$ și deci între ele există relația

$$0 < UFL < \epsilon_{mach} < OFL.$$

Fie x un număr real aproximat de $fl(x) = (\alpha_0 + \frac{\alpha_1}{b} + \frac{\alpha_2}{b^2} + \dots + \frac{\alpha_{p-1}}{b^{p-1}}) b^E$.

Exponentul E poate lua atât valori pozitive cât și valori negative. Cel mai adesea exponentul este "decalat" și reprezentat ca un număr întreg pozitiv (fără semn). Aceasta deoarece ordinea lexicografică (stabilită între șirurile de cifre din reprezentare) și ordinea naturală sunt compatibile în cazul numerelor întregi fără semn. În consecință compararea exponenților (și a numerelor reale corespunzătoare) poate fi făcută eficient. Astfel reprezentarea internă a unui

număr real x aproximat prin $fl(x) = (\alpha_0 + \frac{\alpha_1}{b} + \frac{\alpha_2}{b^2} + \dots + \frac{\alpha_{p-1}}{b^{p-1}}) b^E$ se face sub forma

s	ed	$\alpha_0\alpha_1\dots\alpha_{p-1}$
---	----	-------------------------------------

unde s este semnul lui x (se completează cu 0 dacă semnul este + și cu 1 dacă semnul este -), iar ed este exponentul obținut prin adunarea unui decalaj D la exponentul E :

$$ed = E + D.$$

Standardul IEEE-754

IEEE este acronim pentru Institute of Electrical and Electronics Engineers, o organizație ce are drept principal scop elaborarea standardelor pentru produsele hardware și software. Standardul IEEE-754 se referă la aritmetica în virgulă mobilă în sistemele binare. Acest standard precizează formatul de reprezentare în memorie în simplă și dublă precizie a unui număr real. Reprezentarea se face în virgulă mobilă normalizată:

$$x \approx fl(x) = (1 + \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{p-1}}{2^{p-1}}) 2^E, p = 24, 53.$$

Sunt admise și așa numitele numere denormalizate ("denormalized floating-point numbers"):

$$(0 + \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{p-1}}{2^{p-1}}) 2^E, p = 24, 53,$$

cu cel puțin una dintre cifrele binare $\alpha_1, \alpha_2, \dots, \alpha_{p-1}$ nenule.

Standardul IEEE-754 definește două valori speciale pentru situații excepționale:

- Inf, pe post de "infinit" ("infinity"), pentru rezultatul împărțirii unui număr finit la zero.

- NaN, pe post de "non-număr" ("not a number"), pentru rezultatul următoarelor operații

-]] Adunare : Inf + (-Inf)
-]] Înmulțire: 0·Inf
-]] Împărțire: 0/0 sau Inf/Inf
-]] Calculul restul împărțirii unui număr x la 0 sau a lui Inf la x
-]] Calculul rădăcinii pătrate \sqrt{x} pentru $x < 0$.

Scopul acestor valori este acela de a permite continuarea calculului.

Un număr în virgulă mobilă $(\alpha_0 + \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{p-1}}{2^{p-1}}) 2^E$ se reprezintă intern conform IEEE-754 sub forma

s	ed	$\alpha_1 \alpha_2 \dots \alpha_{p-1}$
---	----	--

unde pentru s se rezervă un bit ce se completează cu 0 dacă numărul este pozitiv și cu 1 dacă numărul este negativ, iar pentru exponentul decalat ed se rezervă k biți (k=8, 11). Decalajul considerat este $D = 2^{k-1} - 1$, deci

$$ed = E + 2^{k-1} - 1,$$

Pe k biți se pot reprezenta ca numere întregi fără semn 2^k valori, de la 0 la $2^k - 1$. Valorile 0 și $2^k - 1$ sunt rezervate pentru numerele denormalizate și pentru valorile speciale Inf și Nan. Deci pentru un număr în virgulă mobilă normalizată trebuie îndeplinită condiția $1 \leq ed \leq 2^k - 2$. De aici rezultă că $-2^{k-1} + 2 \leq E \leq 2^{k-1} - 1$. De exemplu, pe k = 8 biți se pot reprezenta numere întregi fără semn de la 0 la 255. Decalajul considerat este $2^7 - 1 = 127$, deci exponentul E ia valori de la -126 la 127. Numărul de biți rezervați pentru exponent determină intervalul de numere reale reprezentabile în calculator. Numărul de biți rezervați pentru mantisă determină precizia de reprezentare

(gradul de detaliere) a numerelor. Reprezentarea $(\alpha_0 + \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{p-1}}{2^{p-1}})$

$) 2^E$ fiind normalizată, există siguranța că $\alpha_0 = 1$, ceea ce permite omiterea sa (**bit ascuns**) pentru creșterea preciziei de reprezentare, dar complică prelucrarea informației.

Formatele de reprezentare a numerelor în virgulă mobilă (conform standardului IEEE 754) sunt:

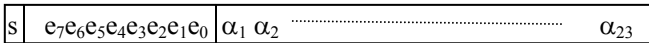
- **simplă precizie (single-precision)** pe 32 de biți:
 - 1 bit pentru semnul mantisei
 - 8 biți pentru exponentul decalat ($E_{\min} = -126, E_{\max} = 127$)
 - 23 biți pentru mantisă ($p = 24, \alpha_0 = 1$ se omite)

- **dublă precizie (double-precision)** pe 64 de biți
 - 1 bit pentru semnul mantisei
 - 11 biți pentru exponentul decalat ($E_{\min} = -1022$, $E_{\max} = 1023$)
 - 52 biți pentru mantisă ($p = 53$, $\alpha_0 = 1$ se omite)

Regula de rotunjire este rotunjirea la par. Deci pentru

- simplă precizie, $\varepsilon_{\text{mach}} = 2^{-24} \approx 10^{-7}$ (7 cifre zecimale semnificative).
- dublă precizie, $\varepsilon_{\text{mach}} = 2^{-53} \approx 10^{-16}$ (16 cifre zecimale semnificative).

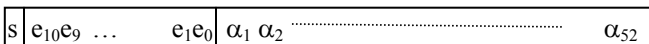
Considerăm o reprezentare în memorie, în simplă precizie:



Fie $ed = e_0 + e_1 2 + e_2 2^2 + \dots + e_7 2^7$ și $m = \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{23}}{2^{23}}$. Valoarea v reprezentată se determină după cum urmează:

-] dacă $0 < ed < 255$, atunci $v = (-1)^s \cdot \overline{1,23} \cdot 2^{\text{ed} - 127}$.
-] dacă $ed = 0$, $\alpha_k = 0$ pentru orice $k = \overline{1,23}$ și $s = 0$, atunci $v = 0$.
-] dacă $ed = 0$, $\alpha_k = 0$ pentru orice $k = \overline{1,23}$ și $s = 1$, atunci $v = -0$.
-] dacă $ed = 0$ și există $\alpha_k \neq 0$, atunci $v = (-1)^s \cdot m \cdot 2^{-126}$; v este o valoare denormalizată
-] dacă $ed = 255$, $\alpha_k = 0$ pentru orice $k = \overline{1,23}$ și $s = 0$, atunci $v = \text{Inf}$.
-] dacă $ed = 255$, $\alpha_k = 0$ pentru orice $k = \overline{1,23}$ și $s = 1$, atunci $v = -\text{Inf}$.
-] dacă $ed = 255$ și există $\alpha_k \neq 0$, atunci $v = \text{NaN}$.

Fie reprezentarea în memorie, în dublă precizie:



Fie $ed = e_0 + e_1 2 + e_2 2^2 + \dots + e_{10} 2^{10}$ și $m = \frac{\alpha_1}{2} + \frac{\alpha_2}{2^2} + \dots + \frac{\alpha_{52}}{2^{52}}$. Valoarea v reprezentată se determină după cum urmează:

-] dacă $0 < ed < 2047$, atunci $v = (-1)^s \cdot (1 + m) \cdot 2^{\text{ed} - 1023}$.

-]] dacă $ed = 0$, $\alpha_k = 0$ pentru orice $k = \overline{1,52}$ și $s = 0$, atunci $v = 0$.
-]] dacă $ed = 0$, $\alpha_k = 0$ pentru orice $k = \overline{1,52}$ și $s = 1$, atunci $v = -0$.
-]] dacă $ed = 0$ și există $\alpha_k \neq 0$, atunci $v = (-1)^s \cdot m \cdot 2^{-1022}$; v este o valoare denormalizată
-]] dacă $ed = 2047$, $\alpha_k = 0$ pentru orice $k = \overline{1,52}$ și $s = 0$, atunci $v = \text{Inf}$.
-]] dacă $ed = 2047$, $\alpha_k = 0$ pentru orice $k = \overline{1,52}$ și $s = 1$, atunci $v = -\text{Inf}$.
-]] dacă $ed = 2047$ și există $\alpha_k \neq 0$, atunci $v = \text{NaN}$.

Exemple:

Să se reprezinte în simplă precizie numerele: $228,15 - 27,25 \quad 0,1 \quad 1,2$
 $x = 228,15$

$$x = 228 + 0,15$$

$$228 = 128 + 64 + 32 + 4 = 2^7 + 2^6 + 2^5 + 2^2 = 11100100_2$$

$$0,15 \cdot 2 = 0,30 = 0 + 0,3$$

$$0,3 \cdot 2 = 0,6 = 0 + 0,6 \quad \left. \begin{array}{l} \\ \\ \\ \\ \end{array} \right\}$$

$$0,6 \cdot 2 = 1,2 = 1 + 0,2$$

$$0,2 \cdot 2 = 0,4 = 0 + 0,4$$

$$0,4 \cdot 2 = 0,8 = 0 + 0,8$$

$$0,8 \cdot 2 = 1,6 = 1 + 0,6 \quad \left. \begin{array}{l} \\ \\ \\ \\ \end{array} \right\}$$

⋮

$$x = 11100100,00100110011001\dots$$

Forma normalizată: $x = 0,111001000010011001\dots \cdot 2^8 = 1,11001000010011001\dots \cdot 2^7$

$$ed = 7 + 2^{8-1} - 1 = 135, ed_2 = 10000110_2$$

$m = 11001000010011001100110$ [011] (am omis primul bit =1, iar cei trei biți din paranteză sunt utilizați pentru rotunjire la par)

$$fl(x) = 1, 11001000010011001100110 \cdot 2^8$$

Reprezentare în virgulă mobilă, simplă precizie, (cu bit ascuns) pentru 228,15:

0	100	001	0	110	0100	0010	0110	0110	0110
4	3	6	4	2	6	6	6	6	6

Deci reprezentării cu bit ascuns a lui 228,15 îi corespunde 43642666 în hexazecimal.

$$x = -27,25$$

$$|x| = 27 + 0,25$$

$$27 = 16 + 8 + 2 + 1 = 2^4 + 2^3 + 2^1 + 2^0 = 11011_2$$

$f_l(x) = 1,00110011001100110011010 \cdot 2^0$ (după cei 23 de biți ai mantisei urmează 100, deci rotunjirea se face astfel încât ultimul bit să aibă valoare pară).

$ed = 0 + 2^{8-1} - 1 = 127 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1$, $ed_2 = 1111111_2$

Reprezentare în virgulă mobilă, simplă precizie (cu bit ascuns) pentru 1,2:

0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

3 F 9 9 9 9 9 A

Deci în reprezentării cu bit ascuns a lui 1,2 îi corespunde 3F99999A în hexazecimal.

Următorul program în C verifică reprezentările de mai sus.

```
#include <stdio.h>
#include <conio.h>
void main() {
    long int *i;
    float f1=228.15, f2=-27.25, f3=0.1, f4=1.2;
    clrscr();
    i=(long int*) &f1;
    printf("\nNumar in virgula mobila:%f\n\tFormat
intern %08lX (hexazecimal)", f1, *i);
    i=(long int*) &f2;
    printf("\nNumar in virgula mobila:%f\n\tFormat
intern %08lX (hexazecimal)", f2, *i);
    i=(long int*) &f3;
    printf("\nNumar in virgula mobila:%f\n\tFormat
intern %08lX (hexazecimal)", f3, *i);
    i=(long int*) &f4;
    printf("\nNumar in virgula mobila:%f\n\tFormat
intern %08lX (hexazecimal)", f4, *i);
    getch();
}
```

Programul afișează

```
Numar in virgula mobila: 228.149994
    Format intern 43642666 (hexazecimal)
Numar in virgula mobila: -27.250000
    Format intern C1DA0000 (hexazecimal)
Numar in virgula mobila: 0.100000
    Format intern 3DCCCCCD (hexazecimal)
Numar in virgula mobila: 1.200000
    Format intern 3F99999A (hexazecimal)
```

III.3. Erori care apar ca urmare a limitelor de reprezentare.

Din secțiunea precedentă rezultă că nu toate numerele reale pot fi reprezentate exact într-un sistem în virgulă mobilă. De asemenea în urma evaluării unei expresii ai cărei operanzi sunt reprezentabili rezultatul obținut nu este neapărat reprezentabil. În mod ideal

$$x \text{ flop } y = \text{fl}(x \text{ op } y)$$

unde op este un operator binar (+, -, *, /), iar flop desemnează corespondentul operatorului respectiv în aritmetica în virgulă mobilă. Sistemele ce satisfac standardul IEEE ating acest ideal în situația în care x op y se găsește în intervalul de numere reale reprezentabile [UFL, OFL]. Depășirea superioară de capacitate (overflow) cauzează de obicei probleme mai serioase decât depășirea inferioară de capacitate (underflow), deoarece nu există nici o aproximație bună pentru un număr real oarecare "mare". Un număr real foarte mic poate fi în mod rezonabil aproximat cu zero. Pe multe sisteme de calcul depășirea superioară de capacitate este fatală, în timp ce în caz de depășire inferioară de capacitate, numărul respectiv este asociat cu zero.

Anumite legi ale aritmeticii reale nu sunt valabile într-un sistem în virgulă mobilă. Astfel adunarea și înmulțirea în virgulă mobilă sunt comutative, dar nu asociative. De exemplu, dacă ε este un număr pozitiv mai mic decât $\varepsilon_{\text{mach}}$, dar mai mare decât $\varepsilon_{\text{mach}}/2$, atunci

$$(1 + \varepsilon) + \varepsilon = 1, \text{ iar } 1 + (\varepsilon + \varepsilon) > 1.$$

În cazul scăderii a două numere reale x și y, poate apărea următorul fenomen (*catastrophic cancellation*)

$$\left| \frac{(\text{fl}(x) - \text{fl}(y)) - \text{fl}(x - y)}{\text{fl}(x - y)} \right| \gg \varepsilon_{\text{mach}},$$

dacă fl(x) este egal (sau foarte apropiat de) fl(y). În următorul program (în C) aproximăm sin(x) printr-o sumă parțială a seriei

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Seria fiind alternantă și convergentă, o sumă parțială de ordin n, aproximează

suma seriei (i.e. sin(x)) cu o eroare absolută maximă de $\frac{|x|^{2n+1}}{(2n+1)!}$.

```
#include<stdio.h>
#include<conio.h>
```

```
#include<math.h>
void main() {
    float x,s,t,eps,x2;
    int i,n;
    clrscr();
    printf("x=");scanf("%f",&x);
    printf("Eroarea=");scanf("%f",&eps);
    t=x;s=0;i=1; x2=x*x;
    while (fabs(t)>=eps){
        s+=t;printf("\n%f",s);
        t=-t*(x2/(4*i*i+2*i));
        i++;
    }
    printf("\nsin(%f) = %f",x,s);
    printf("\nsin(%f) = %f" ,x,sin(x));
    getch();
}
```

Pentru $x=2$ și eroare 10^{-7} se obține aproximația 0.909297 corectă a lui $\sin(2)$.
 Pentru $x = 40$ și eroare 10^{-7} se obține aproximația 523443136.0 a lui $\sin(40)$!
 Valoarea corectă este 0.745113...Acest rezultat se datorează fenomenului de reducere (catastrophic cancellation).

Programul de mai jos reprezintă versiunea în Pascal pentru calculul aproximativ al lui $\sin(x)$:

```
var x,s,t,eps,x2:real;
    i:integer;
begin
    write('x=');readln(x);
    write('Eroare=');readln(eps);
    s:=0;t:=x;x2:=x*x;i:=1;
    while abs(t)>=eps do
        begin
            s:=s+t;
            writeln(i,' ',s);
            t:=-t*(x2/(4*i*i+2*i));
            i:=i+1;
        end;
    writeln('sin(',x,')=',s);
    writeln('sin(',x,')=',sin(x));
    readln
end.
```

Probleme propuse

- 1) Dați reprezentarea internă în simplă precizie a numărului $x = +7,5$.
- 2) Găsiți numărul mașină ce corespunde reprezentării interne în simplă precizie A1E51000 (în hexazecimal).
- 3) Scrieți un program ce calculează sume parțiale ale seriei armonice

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

Observați că în aritmetica virgulei mobile această serie este convergentă! (Este cunoscut că seria armonică este divergentă). Posibile explicații ale faptului că sumele parțiale

$$\sum_{k=1}^n \frac{1}{k}$$

determină un șir staționar în aritmetica virgulei mobile sunt:

- are loc o depășire superioară de capacitate la evaluarea unei sume parțiale
- are loc o depășire inferioară de capacitate la evaluarea lui $1/n$.
- sumele parțiale nu se mai modifică în momentul în care $1/n$ devine neglijabil relativ la suma parțială

$$\frac{1}{n} < \varepsilon_{\text{mach}} \sum_{k=1}^{n-1} \frac{1}{k}$$