Övningar MMG410: flyttalsaritmetik

Larisa Beilina, e-mail: larisa.beilina@chalmers.se

L. Beilina
Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE-412 96 Gothenburg, Sweden, e-mail: larisa.beilina@chalmers.se

1. Vi har ett flyttalsystem med basen $\beta = 10$, precision $t = 4$, och exponentomfång $L = -10$ och $U = 10$. Vilket är det största respektive minsta positiva talet i detta system?
   a) Om systemet är normaliserat?
   b) Om vi tillåter denormaliserade tal?

   Lösning:

   Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

   var

$$0 \leq d_k \leq \beta - 1, L \leq e \leq U,$$

   Största talet i båda fallen är: $\underbrace{9.999}_{4\ siffror} \cdot 10^{10}$.

   Minsta normaliserade talet är $\underbrace{1.000}_{4\ siffror} \cdot 10^{-10}$ och det minsta denormaliserade talet är $\underbrace{0.001}_{4\ siffror} \cdot 10^{-10}$.

   Denormaliserade flyttal används bara (i IEEE-standarden) kring nollan och inte kring största/minsta tal. Det är därför det största talet i b)-uppgiften är samma som i a)-uppgiften. De största talet är i båda deluppgifterna normaliserat.
2. Vi har ett flyttalsystem med basen 10, precision t och exponentomfång $[L,U]$. a) Vilka är de minsta värdena på t, U och det största på L så att både 2365.27 och 0.0000512 kan representeras exakt i normaliserad form? b) Om vi tillåter denormaliserade tal (denormaliserade eller subnormala tal: offra "decimaler" för att få större exponentomfång) ?
   Lösning:
   Flyttal (tal med flytande decimalpunkt):

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + ... + \frac{d_{t-1}}{\beta^{t-1}} \right) \beta^e,$$

   var

$$0 \leq d_k \leq \beta - 1, L \leq e \leq U,$$

   a) För 2365.27 normaliserat blir $\underbrace{2.36527}_{6\ siffror} \cdot 10^3$ med $t = 6$ och $U = 3$. För $0.0000512 = 5.12 \cdot 10^{-5}$ så $L = -5$.
   b) Med $t = 6$ får vi $0.0000512 = \underbrace{0.00512}_{6\ siffror} \cdot 10^{-2}$ så $L = -2$.
3. Skriv talet -3.25 i binär (bas 2) och i hexadecimalt (bas 16) form, som flyttal i dator.

Lösning:

$$-3.25: \ -[1.625]\cdot 2^1 = -\underbrace{[1+0.625]}_{mantissa}\cdot 2^1$$

Exponenten $e = 1$ lagras som: $1+1023 = 1024 = 2^{10}$. Mantissa: 1 kodas inte,

$$0.625 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \ldots = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 0 + \frac{1}{8}\cdot 1 + \ldots$$

$\frac{1}{2} = 0.5 < 0.625; \quad \frac{1}{4} = 0.25 : 0.625 - 0.5 = 0.125; 0.25 > 0.125$, darför $\frac{1}{4}\cdot 0$,
$\frac{1}{8} = 0.125 = 0.125$ och darför $\frac{1}{8}\cdot 1$, resten i mantissa ska vara 0.
Vi får följande binär (bas 2) representation för -3.25 :

1	10000000000	1010 0
tecken	exponent 11 bitar	mantissa 52 bitar

Bas 16:

0 1 2 3 4 5 6 7 8 9	10	11	12	13	14	15
	a	b	c	d	e	f

Nu grupperar vi om binär form för -3.25 i 4 bitar:

1100|0000|0000|1010|0000 0000

och kodar första fyra bitar:

1100 = c
eftersom
$1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 12 = c$
0000 koderas som 0, och sedan

1010 = a
eftersom

$1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 10 = a.$
Slutsats: -3.25 i hexadecimalt (bas 16) form lagras som:

c00a000000000000

4. Skriv talet -9.28 i binär (bas 2) form som flyttal i dator.

Lösning:

$-9.28 := -[1.16]\cdot 2^3 = -[1+0.16]\cdot 2^3$
Exponenten 3 lagras som:

$$3+1023 = 1026 = 1024+2 = 1\cdot 2^{10} + 1\cdot 2^1 + 0\cdot 2^0$$

I mantissan 1 kodas inte, vi kodar bara 0.16:

$$0.16 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{16}x_4 + \frac{1}{32}x_5 + \ldots =$$
$$= \frac{1}{2}\cdot 0 + \frac{1}{4}\cdot 0 + \frac{1}{8}\cdot 1 + \frac{1}{16}\cdot 0 + \ldots$$

Förklaringen hur kodas mantissa:
$\frac{1}{2} = 0.5 > 0.16$, darför $\frac{1}{2}\cdot 0$, $\frac{1}{4} = 0.25 : 0.25 > 0.16$, darför $\frac{1}{4}\cdot 0$, $\frac{1}{8} = 0.125 <$ 0.16 och darför $\frac{1}{8}\cdot 1$, $\frac{1}{16} = 0.0625 : 0.16 - 0.125 = 0.035, 0.0625 > 0.035$, och darför $\frac{1}{16}\cdot 0$, $\frac{1}{32} = 0.0312 : 0.0312 < 0.035$, och darför $\frac{1}{32}\cdot 1$, och så vidare ...
Vi observerar att det är inte tillräkligt 52 bitar i mantissan för att koda exakt 0.16. I andra ord, 0.16 kan inte presenteras exakt i binär form, darför binär form introducerar fel i lagringen av flyttal.

1	10000000010	00101 ....
tecken	exponent 11 bitar	mantissa 52 bitar

Kollar i Matlab:

```
q = quantizer('double');

y = num2bin(q,-9.28)

y =

1100000000100010100011110101110000101000111101011100001010001111
```

5. Skriv talet 1 i binär (bas 2) form som flyttal i dator.

Lösning:

$1 := [1]\cdot 2^0 = [1+0.0]\cdot 2^0$
Mantissa är 0 här.
Exponenten är också 0 och lagras som:
$0+1023 = 1024-1 = 1\cdot 2^{10} - 1\cdot 2^0 = \underbrace{10000000000}_{11\ bitar} - \underbrace{00000000001}_{11\ bitar} = \underbrace{01111111111}_{11\ bitar}$.

0	01111111111	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

6. Skriv talet 0.5 i binär (bas 2) form som flyttal i dator.

Lösning:

$0.5 := [1]\cdot 2^{-1} = [1+0.0]\cdot 2^{-1}$

Mantissa är 0 här.
Exponenten -1 lagras som:
$-1+1023 = 1024-2 = 1\cdot 2^{10} - 1\cdot 2^1 = \underbrace{10000000000}_{11\ bitar} - \underbrace{00000000010}_{11\ bitar} = \underbrace{01111111110}_{11\ bitar}$.

0	01111111110	0000 ....
tecken	exponent 11 bitar	mantissa 52 bitar

7. Skriv talet 0.1 i binär (bas 2) form som flyttal i dator.

Lösning:

$0.1 := [1.6]\cdot 2^{-4} = [1+0.6]\cdot 2^{-4}$
Mantissa är 0.6 här. Det är inte tillräkligt 52 bitar i mantissan för att koda exakt 0.6, se nedan Matlab's presentation. Det betyder att 0.6 presenteras inte exakt i binär form.
Exponenten -4 lagras som:
$-4+1023 = 1019 = 1024-5 = 1\cdot 2^{10} - (1\cdot 2^2 + 1\cdot 2^0) = \underbrace{10000000000}_{11\ bitar} - \underbrace{00000000101}_{11\ bitar} =$

$\underbrace{01111111011}_{11\ bitar}$.

0	01111111011	100...
tecken	exponent 11 bitar	mantissa 52 bitar

Kollar i Matlab:

```
    q = quantizer('double');

y = num2bin(q,0.1)


  y =

  0011111110111001100110011001100110011001100110011001100110011010
```

8. Antag att vi arbetar med fyrsiffrig decimal aritmetik. Beräkna följande summor samt de absoluta och relativa felen:

- 6.278 + 4.039
- 6.278e10 + 4.039e10
- 6.278e-10 + 4.039e-10

Lösning:

Vi kan klara av alla tre fallen på en gång genom att låta p, nedan, anta värdena $p = 0, 10, -10$. Det exakta värdet blir $x = 6.278\cdot 10^p + 4.039\cdot 10^p = 1.0317\cdot 10^{p+1}$ och det lagras, korrekt avrundat och i normaliserad form, som $\hat{x} = 1.032\cdot 10^{p+1}$. Det absoluta felet blir:

- Det absoluta felet blir: $|x-\hat{x}| = |1.0317 \cdot 10^{p+1} - 1.032 \cdot 10^{p+1}| = 3 \cdot 10^{-4} \cdot 10^{p+1} = 3 \cdot 10^{p-3}$.
- Det relativa felet blir: $\frac{|x-\hat{x}|}{|x|} = 3 \cdot 10^{p-3} / 1.0317 \cdot 10^{p+1} \approx 3 \cdot 10^{-4}$.

Notera att de tre absoluta felen blir: $3 \cdot 10^{-3}$, $3 \cdot 10^{7}, 3 \cdot 10^{-13}$. Det relativa felet blir, i alla tre fallen, $\approx 3 \cdot 10^{-4}$.

9. Visa att addition enligt IEEE är en stabil algoritm för maskinnoggrannheten ε_{mach}.

Lösning:

Vi vet att

$$fl(a+b) = (a+b)(1+\varepsilon), |\varepsilon| \leq \varepsilon_{mach}.$$

Alltså gäller att

$$\begin{aligned} fl(a+b) &= \tilde{a} + \tilde{b}, \\ \tilde{a} &= a(1+\varepsilon),\ \tilde{b} = b(1+\varepsilon), \\ |\tilde{a}-a| &\leq \varepsilon_{mach}|a|,\ |\tilde{b}-b| \leq \varepsilon_{mach}|b| \end{aligned}$$

fl(a + b) är således den exakta summan av två något störda tal.

10. Är skalärproduktsberäkning med IEEE stabil? Dvs. är det stabilt att bilda

$$\sum_{k=1}^{n} x_k y_k$$

Lösning:

Låt oss betrakta specialfallet då $n = 4$. För att få mindre oläsliga formler inför vi $\sigma_k = 1 + \delta_k$, $\pi_k = 1 + \varepsilon_k$ (σ för summa och π för produkt) där alla $|\varepsilon_k| < \varepsilon_{mach}$ och $|\delta_k| < \varepsilon_{mach}$. Vi har:

- $fl(x_1 y_1) = x_1 y_1 \pi_1$ (vi använder inte σ_1)
- $fl(x_1 y_1 + x_2 y_2) = [x_1 y_1 \pi_1 + x_2 y_2 \pi_2]\sigma_2$
- $f(x_1 y_1 + x_2 y_2 + x_3 y_3) = ([x_1 y_1 \pi_1 + x_2 y_2 \pi_2]\sigma_2 + x_3 y_3 \pi_3)\sigma_3$.
- $fl(x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4) = [([x_1 y_1 \pi_1 + x_2 y_2 \pi_2]\sigma_2 + x_3 y_3 \pi_3)\sigma_3 + x_4 y_4 \pi_4]\sigma_4$.

Allmänt gäller tydligen, om vi inför $\sigma_{j:k} = \sigma_j \sigma_{j+1} \ldots \sigma_k$, att:

$$\begin{aligned} fl(x_1 y_1 + x_2 y_2 + \ldots + x_n y_n) = {} & x_1 y_1 \pi_1 \sigma_{2:n} + x_2 y_2 \pi_2 \sigma_{2:n} + x_3 y_3 \pi_3 \sigma_{3:n} \\ & + \ldots + x_k y_k \pi_k \sigma_{k:n} + \ldots + x_n y_n \pi_n \sigma_n. \end{aligned}$$

Med våra antaganden om flyttalsaritmetik kan vi skriva detta

$$fl(\sum_{k=1}^{n} x_k y_k) = x_1 y_1(1+n\varepsilon_1') + \sum_{k=2}^{n} x_k y_k(1+(n-k+2)\varepsilon_k')$$

där alla $|\varepsilon_k'| \leq \varepsilon_{mach}$. Om vi t.ex. sätter $\tilde{x}_k = x_k\sqrt{1+(n-k+2)\varepsilon_k'}$ och $\tilde{y}_k = y_k\sqrt{1+(n-k+2)\varepsilon_k'}$ så gäller tydligen att:

$$fl(\sum_{k=1}^{n} x_k y_k) = \sum_{k=1}^{n} \tilde{x}_k \tilde{y}_k$$

Om då $n\varepsilon_{mach}$ är tillräckligt litet så är den beräknade skalärprodukten en exakt skalärprodukt av näraliggande värden och algoritmen är stabil.