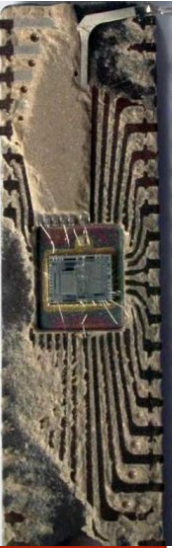


Floating-point numbers

ver. 11 z drobnymi modyfikacjami!

Wojciech Myszka

2023-10-23 15:23:58 +0200

Wrocław University of Science and Technology

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one
2. Replacing arithmetic operations with logical ones

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one
2. Replacing arithmetic operations with logical ones
3. Using two's complement for signed number representation (suggested by von Neumann in 1945, but widely implemented in the early sixties)

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one
2. Replacing arithmetic operations with logical ones
3. Using two's complement for signed number representation (suggested by von Neumann in 1945, but widely implemented in the early sixties)
4. The sign (leftmost bit) is treated as a data in all operations.

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one
2. Replacing arithmetic operations with logical ones
3. Using two's complement for signed number representation (suggested by von Neumann in 1945, but widely implemented in the early sixties)
4. The sign (leftmost bit) is treated as a data in all operations.

Foreword

Main milestones in building modern computers:

1. Binary numerical system instead of decimal one
2. Replacing arithmetic operations with logical ones
3. Using two's complement for signed number representation (suggested by von Neumann in 1945, but widely implemented in the early sixties)
4. The sign (leftmost bit) is treated as a data in all operations.

Problems:

- limited number of bits for storing values
- 8, 16, 32, 64, 128,. . .
- when the result of arithmetic operation does not “fit” — result is wrong! This is called *overflow*.

Part I

Basic arithmetic

How to store non-integer numbers?

In spreadsheet we can choose between:

- general: 1.5782
- number 1.58
- percent 157.82%
- currency 1.58 €
- scientific 1.58E+00
- thousand separator

This is only external representation!
What about internal representation?

Fixed point

Let's assume 16 bits non-integer arithmetic.

1. First eight bits for the integer part
2. Second eight bits for the fractional part

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sign	integer part							fractional part							

Floating-point

All sixteen bits (in general 32 or 64) are used for storing

1. *sign*
2. fractional part x
3. exponent e

and the number is represented as $sign \cdot x \cdot 2^e$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sign	(e) exponent					(x) fractional part									

Why “floating-point”

Fixed-point vs *Floating-point*

- Fixed point means that we are using a fixed number of digits for remembering the integer part of a number and a fixed number for the non-integer (fractional) part, for example, 6 (integer) + 2 (fractional) in “financial calculator” (for home use). So the position of the decimal point is also fixed.
- Floating-point means that position of the decimal point changes during calculations: we have a fixed number of **significant** digits.

Fractions

In decimal numbers we have:
$345.5 = 3 * 10^2 + 4 * 10^1 + 5 * 10^0 + 5 * 10^{-1}$
So, analogically we can write:
$101011001.1_{(2)} = 2^8 + 2^6 + 2^4 + 2^3 + 2^0 + 2^{-1}$

Let's think about 3rd (binary) digit (in fractional part)
$0.001_{(2)} = 0.125_{(10)}$
To store fractions with reasonable precision this needs a lot of bits. The same when there is a lot of digits on the left of "digital point".

Big decimal numbers

To store big number one can use "scientific notation" (or floating-point). Instead of writing

$$c = 299792458 \mathrm{~m/s}$$

one can write

$$c = 2.99792458 * 10^8 \mathrm{~m/s}$$

or

$$c = 2.99792458 \mathrm{~E}\, 8 \mathrm{~m/s}$$

or (approximately)

$$c = 3 \mathrm{~E}\, 8 \mathrm{~m/s}$$

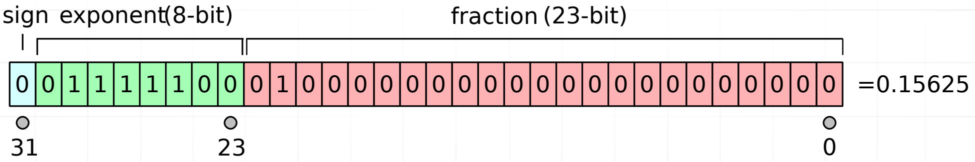
2.99792458 is sometimes called a significand (mantissa); 8 is called an exponent.

"Big" binary digits

One can use the same way

$$1.11_{(2)} * 2^7$$

Floating-point numbers are described in IEEE-754 Standard (established in 1985):



All floating-point numbers are stored in, so-called, *normalized form*, i.e. there is only one digit on the left side of the decimal point, and is greater than zero.

Precision

32 bits numbers

- "Binary" precision: 24 bits
- "Decimal" precision: ≈ 7.2 decimal digits (this means: mostly 7, sometimes 8, in average 7.2).

64 bits numbers

- "Binary" precision: 53 bits
- "Decimal" precision: ≈ 15.9 decimal digits.

Range

1. 32-bits: $1.17549\ \mathrm{E}-038$ to $3.40282\ \mathrm{E}+038$
2. 64-bits: $2.22507\ \mathrm{E}-308$ to $1.79769\ \mathrm{E}+308$

Range

1. 32-bits: $1.17549\,\mathrm{E}-038$ to $3.40282\,\mathrm{E}+038$
2. 64-bits: $2.22507\,\mathrm{E}-308$ to $1.79769\,\mathrm{E}+308$

Floating-point values...

... are **rational numbers**.
They can be expressed as the quotient or fraction $\frac{p}{q}$ of two integers, a numerator p and a non-zero denominator q.

Floating-point arithmetic I

Let's assume that our computer uses decimal floating-point arithmetic with three digits.

1. Multiplication.
   Easy: multiplying of mantissas and adding exponents
   $1.33\,e{+}3 * 1.55\,e{+}7 = 2.0615\,e{+}10$
   Next, the result has to be "shorten" (cut) to three digits: $2.06\,e{+}10$
   Note: We are loosing value of $0.0015\,e{+}10 = 15\,000\,000$ (Yes! Fifteen millions!)
   Caution: Sometimes strange happens. After the operation result is **denormalised**: more then one figure on the left to the digital point. Result is normalized (exponent is corrected), and rounded :
   $5.55\,e{+}0 * 6.33e{+}0 = 35.13\,e{+}0 = 3.51\,e{+}1$
2. Division:
   Like multiplication: mantissas divided, exponent subtracted.
   $1.33\,e{+}0/9.88\,e{+}0 = 0.134615385\,e{+}0 = 1.35\,e{-}1$
   (result normalized and rounded).

Floating-point arithmetic II

3. Addition.
   A simple method to add floating-point numbers is to first represent them with the same exponent (denormalize!)
   $1.22\,e{+}0 + 3.35\,e - 4 = 1.22\,e{+}0 + 0.000335\,e{+}0 = 1.220335\,e{+}0 = 1.22\,e{+}0$
   and next normalize and round...
   But check carefully the result!
4. Subtraction.
   Like addition.

Some problems. . .

1. The limited number of bits used to store numbers! (There are special applications allowing for arithmetic with arbitrary numbers of digits.)
2. "Overflow" occurs when the result of arithmetic operation does not "fit" in a word (32 or 64 bits).
3. Most of the numbers that (out of habit), are considered to be accurate, do not have exact binary representation (0.5 is OK but 0.1 NO).

Some problems. . .

1. The limited number of bits used to store numbers! (There are special applications allowing for arithmetic with arbitrary numbers of digits.)
2. "Overflow" occurs when the result of arithmetic operation does not "fit" in a word (32 or 64 bits).
3. Most of the numbers that (out of habit), are considered to be accurate, do not have exact binary representation (0.5 is OK but 0.1 NO).
4. The last one is OK. We do remember fraction $\frac{1}{3} = 0.3(\bar{3})$, but. . .

Part II

Absolute errors theory

Basic definitions

Quantity

any mathematical constant, the result of some mathematical operations (actions), a root of the equation solved. π is defined as the ratio of the circumference of a circle to its diameter, $\sqrt{2}$ is a root of the quadratic equation $x^2 - 2 = 0$ (or the diagonal of a unit square).

Basic definitions

Quantity

any mathematical constant, the result of some mathematical operations (actions), a root of the equation solved. π is defined as the ratio of the circumference of a circle to its diameter, $\sqrt{2}$ is a root of the quadratic equation $x^2 - 2 = 0$ (or the diagonal of a unit square).

The exact value of the quantity

value derived directly from the definition, not burdened by any errors.

Basic definitions

Quantity

any mathematical constant, the result of some mathematical operations (actions), a root of the equation solved. π is defined as the ratio of the circumference of a circle to its diameter, $\sqrt{2}$ is a root of the quadratic equation $x^2 - 2 = 0$ (or the diagonal of a unit square).

The exact value of the quantity

value derived directly from the definition, not burdened by any errors.

The approximate value of the quantity

numerical value obtained by calculation. Typically, the calculation did not get the exact value.

Physical quantities

Pressure, temperature, length, concentration — are examples of physical quantities, which often are measured.

Each measurement was burdened by an **error** resulting from the **accuracy** of the measurement tool used.

So, for example:

- **Quantity:** the temperature at some point in the room.
- **The exact value of the quantity:** — the temperature at this particular point
- **The approximate value of the quantity:** — temperature measured with a thermometer.

Calculations

- We use a computer to make some calculations.

Calculations

- We use a computer to make some calculations.
- The computer gives the result ($a = 5.34273343$) with 8 digits after the decimal point.

Calculations

- We use a computer to make some calculations.
- The computer gives the result ($a = 5.34273343$) with 8 digits after the decimal point.
- Can we say that the result has all the figures **correct**? I mean that the difference between the result and the exact value Is less than 0.5×10^{-8}?

Calculations

- We use a computer to make some calculations.
- The computer gives the result ($a = 5.34273343$) with 8 digits after the decimal point.
- Can we say that the result has all the figures **correct**? I mean that the difference between the result and the exact value Is less than 0.5×10^{-8}?
- What about the situation that the method of calculation is inaccurate?

Mathematical pendulum

Let's consider the mathematical pendulum.
The equation describing the period of oscillations (in seconds) looks like:

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where L is the length (in meters) and g is a gravitational acceleration constant.

Mathematical pendulum

Let's consider the mathematical pendulum.
The equation describing the period of oscillations (in seconds) looks like:

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where L is the length (in meters) and g is a gravitational acceleration constant.
What is the value of π?

Mathematical pendulum

Let's consider the mathematical pendulum.
The equation describing the period of oscillations (in seconds) looks like:

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where L is the length (in meters) and g is a gravitational acceleration constant.
What is the value of π?
What is the value of g?

Mathematical pendulum

Let's consider the mathematical pendulum.
The equation describing the period of oscillations (in seconds) looks like:

$$T = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

where L is the length (in meters) and g is a gravitational acceleration constant.
What is the value of π?
What is the value of g?
Let's assume, that $L = 1$ m.

Mathematical pendulum

L	g	pi	T	
1	9.81	3.14	2.00504969	
1	9.81	3.142	2.00632679	
1	9.81	3.1416	2.00607137	
1	9.81	3.14159	2.00606499	
1	9.81	3.141593	2.00606690	
1	9.81	3.141592654	2.00606668	
1	9.80665	3.141592654	2.00640929	
1	9.83332	3.141592654	2.00368655	pole
1	9.7803	3.141592654	2.00911030	equatorial
1	9.8115	3.141592654	2.00591333	Wrocław
1	9.8123	3.141592654	2.00583156	Warsaw
1	9.8105	3.141592654	2.00601556	Kraków
1	9.8337	3.141592654	2.00364783	arctic ocean

The absolute error of the approximate (measured) value of the quantity I

Let A be the exact value of the quantity, and a be its approximate value.

Absolute error

is any number Δa satisfying the condition:

$$|A - a| \leq \Delta a,$$

i.e. such a number, that:

$$a - \Delta a \leq A \leq a + \Delta a.$$

Approximate value a and its absolute error Δa determine an interval:

$$< a - \Delta a; a + \Delta a >,$$

to which belongs the exact value A.

The absolute error is not specified unambiguously!

Rough value

Rough value

If a is an approximate value of quantity A with an error Δa then

$$\frac{\Delta a}{a}$$

I will call **rough value** for A.

π

We know that $\pi = 3.14159265\ldots$. 3.14 is commonly used as an approximation of π in calculations.
What is an absolute error of this approximation?

π

We know that $\pi = 3.14159265\ldots$. 3.14 is commonly used as an approximation of π in calculations.
What is an absolute error of this approximation?
Because

$$3.14 - 0.0016 \leq \pi \leq 3.14 + 0.0016$$

or

$$3.1384 \leq \pi \leq 3.1416$$

the absolute error of this approximation is 0.0016.
Exact value of π is somewhere in the interval

$$< 3.1384; 3.1416 >$$

so we can write $\pi = \overset{0.0016}{3.14}$

“Approximate equality”

If two rough values $\overset{\alpha}{a}$ and $\overset{\beta}{b}$ are such, that the interval $< a - \alpha; a + \alpha >$ is **included** in the interval $< b - \beta; b + \beta >$ we can tell that $\overset{\alpha}{a}$ is approximately equal to $\overset{\beta}{b}$. We will note this as:

$$\overset{\alpha}{a} \Rightarrow \overset{\beta}{b}$$

The fact that $\overset{\alpha}{a}$ is approximately equal to $\overset{\beta}{b}$ does not imply the opposite: that $\overset{\beta}{b}$ is approximately equal to $\overset{\alpha}{a}$! (The relation is not reflexive.)

Rounding rough values

For any rough value $\overset{\alpha}{a}$ and any real value b following relation is fulfilled:

$$\overset{\alpha}{a} \Rightarrow \overset{\alpha+|a-b|}{b}$$

this means that $\overset{\alpha}{a}$ is approximately equal to $\overset{\alpha+|a-b|}{b}$

Rounding rough values

For any rough value $\overset{\alpha}{a}$ and any real value b following relation is fulfilled:

$$\overset{\alpha}{a} \Rightarrow \overset{\alpha+|a-b|}{b}$$

this means that $\overset{\alpha}{a}$ is approximately equal to $\overset{\alpha+|a-b|}{b}$

So, instead of using $\overset{0.000000003}{3.14159265}$ as the π I can use simply 3 but...

Rounding rough values

For any rough value $\overset{\alpha}{a}$ and any real value b following relation is fulfilled:

$$\overset{\alpha}{a} \Rightarrow \overset{\alpha+|a-b|}{b}$$

this means that $\overset{\alpha}{a}$ is approximately equal to $\overset{\alpha+|a-b|}{b}$

So, instead of using $\overset{0.000000003}{3.14159265}$ as the π I can use simply 3 but...

...I should tell that the error is $|3.14159265 - 3| + 0.000000003 = 0.141592653$!

$$\overset{0.000000003}{3.14159265} \Rightarrow \overset{0.141592653}{3}$$

Rounding rough values

For any rough value $\overset{\alpha}{a}$ and any real value b following relation is fulfilled:

$$\overset{\alpha}{a} \Rightarrow \overset{\alpha+|a-b|}{b}$$

this means that $\overset{\alpha}{a}$ is approximately equal to $\overset{\alpha+|a-b|}{b}$

So, instead of using $\overset{0.000000003}{3.14159265}$ as the π I can use simply 3 but...

...I should tell that the error is $|3.14159265 - 3| + 0.000000003 = 0.141592653$!

$$\overset{0.000000003}{3.14159265} \Rightarrow \overset{0.141592653}{3}$$

Calculations are simpler but burdened with 47 197 551 times greater absolute error.

Rounding rough values cont.

Of course, the following rounding is "better":

$$\overset{0.0000027}{3.14159} \Rightarrow \overset{0.0015927}{3.14} \Rightarrow \overset{0.0016}{3.14}$$

We are rounding numbers for practical reasons: when the result has too much significant digits. . .

Rounding rough values cont.

When $a = b$ and $\beta \geq \alpha$, we can write that:

$$\overset{\alpha}{a} \Rightarrow \overset{\beta}{b}$$

So $\overset{0.0015927}{3.14} \Rightarrow \overset{0.0016}{3.14}$
These values are approximately equal.

Rounding rules I

- When the result of calculations has a lot of digits we can remove (cut off) the least significant digits — increasing rounding error.
- When the first of removed digits is 0, 1, 2, 3, 4 — the last remaining digit is not changed.
- When the first of removed digits is 5, 6, 7, 8, 9 — we are increasing the result by one on the least significant digit.

These rules are sometimes called "proper rounding," but easily one can find other rules:

- Round half up
- Round half down
- Round half away from zero
- Round half towards zero
- Round half to even (banker's rounding)

Rounding rules II

- Round half to odd
- Stochastic rounding

Homework: Read about rounding rules and decide which one (of the above mentioned rules) is “proper rounding?”

Arithmetic operations on a rough values

addition

$$\overset{\alpha}{a} + \overset{\beta}{b} = \overset{\alpha+\beta}{a+b}$$

Arithmetic operations on a rough values

addition

$$\overset{\alpha}{a} + \overset{\beta}{b} = \overset{\alpha+\beta}{a + b}$$

subtraction

$$\overset{\alpha}{a} - \overset{\beta}{b} = \overset{\alpha+\beta}{a - b}$$

Arithmetic operations on a rough values

multiplication

$$\overset{\alpha}{a} \cdot \overset{\beta}{b} \Rightarrow \overset{|a|\beta+|b|\alpha+\alpha\beta}{ab}$$

Arithmetic operations on a rough values

multiplication

$$\overset{\alpha}{a} \cdot \overset{\beta}{b} \Rightarrow \overset{|a|\beta+|b|\alpha+\alpha\beta}{ab}$$

division

$$\overset{\alpha}{a} : \overset{\beta}{b} \Rightarrow \frac{\overset{\gamma}{a}}{b}$$

where

$$\gamma = \frac{\alpha + |\frac{a}{b}|\beta}{|b| - \beta}.$$

Arithmetic operations on a rough values

Homework

Error in other mathematical operations:

1. rising to power?
2. square root?
3. trigonometric functions?
4. other. . . ?

Arithmetic operations on rough values

addition

Do we need to memorize all these rules? Better is to understand them.

1. The first "worst case":

$$a - \alpha + b - \beta = (a + b) - (\alpha + \beta)$$

2. The second "worst case":

$$a + \alpha + b + \beta = (a + b) + (\alpha + \beta)$$

Arithmetic operations on rough values

addition

Do we need to memorize all these rules? Better is to understand them.

1. The first "worst case":

$$a - \alpha + b - \beta = (a + b) - (\alpha + \beta)$$

2. The second "worst case":

$$a + \alpha + b + \beta = (a + b) + (\alpha + \beta)$$

(Homework: How it will work for subtraction? Multiplication? Division?)

Example

Calculate the value of the polynomial

$$w(x) = a_0x^4 + a_1x^3 + a_2x^2 + a_3x + a_4$$

for $x = 2.1$.

Let's assume, those polynomial coefficients are exact values, and are equal to:

$$a_0 = 2.3,\ a_1 = 3,\ a_2 = -4.5,\ a_3 = 7.2,\ a_4 = -0.1$$

First, we perform calculations accurate to two decimal places, and then to four.

Example, cont. I

two decimal places

$$x^2 = \overset{0.0}{2.1} \times \overset{0.0}{2.1} = \overset{0.00}{4.41}$$

$$x^3 = \overset{0.00}{4.41} \times \overset{0.0}{2.1} = \overset{0.000}{9.261} \Rightarrow \overset{0.001}{9.26}$$

$$x^4 = \overset{0.001}{9.26} \times \overset{0.0}{2.1} \Rightarrow \overset{0.0021}{19.446} \Rightarrow \overset{0.0061}{19.45}$$

$$2.3 \times x^4 = \overset{0.0}{2.3} \times \overset{0.0061}{19.45} \Rightarrow \overset{0.01403}{44.735} \Rightarrow \overset{0.01903}{44.74} \Rightarrow \overset{0.02}{44.74}$$

$$3x^3 = \overset{0}{3} \times \overset{0.001}{9.26} \Rightarrow \overset{0.003}{27.78}$$

$$-4.5x^2 = \overset{0.0}{-4.5} \times \overset{0.00}{4.41} \Rightarrow \overset{0.000}{-19.845} \Rightarrow \overset{0.005}{-19.85}$$

$$7.2x = \overset{0.0}{7.2} \times \overset{0.0}{2.1} \Rightarrow \overset{0.00}{15.12}$$

sum:

$$w(2.1) = \overset{0.02}{44.74} + \overset{0.003}{27.78} \overset{0.005}{-19.85} + \overset{0.00}{15.12} \overset{0.0}{-0.1} = \overset{0.028}{67.69}$$

Example, cont. I

four decimal places

$$x^2 = \overset{0.0}{2.1} \times \overset{0.0}{2.1} = \overset{0.00}{4.41}$$

$$x^3 = \overset{0.00}{4.41} \times \overset{0.0}{2.1} = \overset{0.000}{9.261}$$

$$x^4 = \overset{0.000}{9.261} \times \overset{0.0}{2.1} = \overset{0.0000}{19.4481}$$

$$2.3 \times x^4 = \overset{0.0}{2.3} \times \overset{0.0000}{19.4481} = \overset{0.0000}{44.73063} \Rightarrow \overset{0.00003}{44.7306}$$

$$3x^3 = \overset{0}{3} \times \overset{0.000}{9.261} = \overset{0.000}{27.783}$$

$$-4.5x^2 = \overset{0.0}{-4.5} \times \overset{0.00}{4.41} = \overset{0.000}{-19.845}$$

$$7.2x = \overset{0.0}{7.2} \times \overset{0.0}{2.1} = \overset{0.00}{15.12}$$

sum

$$w(2.1) = \overset{0.00003}{44.7306} + \overset{0.000}{27.783} - \overset{0.000}{19.845} + \overset{0.00}{15.12} - \overset{0.0}{0.1} = \overset{0.00003}{67.6886}$$

Example, cont. I

Let's assume now that coefficients are not exact values. There are "rough values" :

$$a_0 \overset{0.01}{=} 2.3,\ a_1 \overset{0}{=} 3,\ a_2 \overset{0.02}{=} -4.5,\ a_3 \overset{0.02}{=} 7.2,\ a_4 \overset{0.01}{=} -0.1$$

two decimal places

$$w(2.1) \Rightarrow \overset{0.42}{67.69}$$

four decimal places

$$w(2.1) \Rightarrow \overset{0.3678}{67.6886} \Rightarrow \overset{0.3692}{67.69} \Rightarrow \overset{0.37}{67.69}$$

Example, cont. II

Conclusion

When **the input data** are not accurate, increasing the number of significant digits (in calculations) does not increase the accuracy of the result.
The rule is known as *Garbage in, garbage out...*

Yet another example

$$x_0 \in [0; 1]$$

$$x_{k+1} = \mu x_k (1 - x_k)$$

Let $\mu = 3.7$

Results

N	32 bit	64 bit	128 bit
2	0.2566875	0.2566875	0.2566875
4	0.7680534	0.7680533	0.7680533
6	0.8312892	0.8312889	0.8312889
8	0.9236761	0.9236760	0.9236760
10	0.7133774	0.7133778	0.7133778
12	0.6814939	0.6814953	0.6814953
14	0.5850334	0.5850375	0.5850375
16	0.3381789	0.3381866	0.3381866
18	0.5266685	0.5266460	0.5266460
20	0.2649377	0.2649240	0.2649240
24	0.7726893	0.7727947	0.7727947
28	0.9247919	0.9247575	0.9247575

N	32 bit	64 bit	128 bit
32	0.6627011	0.6632869	0.6632869
36	0.2665924	0.2677791	0.2677791
40	0.7597211	0.7502111	0.7502111
45	0.3075923	0.4160662	0.4160662
50	0.8943822	0.9210730	0.9210730
55	0.2570739	0.7404139	0.7404139
60	0.5249998	0.5649204	0.5649208
70	0.9157254	0.6021104	0.6020892
80	0.9222577	0.4007202	0.4019857
90	0.2573895	0.5755109	0.6455021
100	0.7139580	0.3158045	0.8947899
110	0.2567323	0.7575933	0.5199780

Results (computed up to 1000 significant digits)

```
  k      x(k)
----  -------
 10: 0.7133778
 20: 0.264924
 30: 0.7073271
 40: 0.7502111
 50: 0.921073
 60: 0.5649208
 70: 0.6020892
 80: 0.401986
 90: 0.6455194
100: 0.8950223
110: 0.5189601
```

Computer program

```
#include <stdio.h>
int main()
{
        float s;
        double d;
        long double e;
        int i;
        s = d = e = 0.5;
        for(i=1;i<=110;i++)
        {
                s = 3.7F * s * (1.F - s);
                d = 3.7* d * (1. - d);
                e = 3.7L * e * (1.L - e);
                if (i%10==0)
                    printf("%10i %.7f %.7lf %.7Lf\n", i, s, d, e);
        }
        return 0;
}
```

Additional literature I

- Zuber R., *Metody numeryczne i programowanie*, WSziP 1975, fragmenty: https://kmim.wm.pwr.edu.pl/myszka/wp-content/uploads/sites/2/2020/10/zuber.pdf.
- Myszka W., *Przykładowe programiki pokazujące problemy numeryczne* 2008, dostępne pod adresem https://kmim.wm.pwr.edu.pl/myszka/wp-content/uploads/sites/2/2020/10/kod.pdf.
- Goldberg D., *What every computer scientist should know about Floating-Point arithmetic*, [w:] *Numerical Computation Guide*, Sun Microsystems, Palo Alto 2000, URL http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.
- Ward A., *Some issues on floating-point precision under linux*, Linux Gazette, , 53, 2000, URL https://linuxgazette.net/issue53/ward.html.

Colophon

Presentation typeset using $\LaTeX\,2_\varepsilon$ system with beamer class using Latin Modern font.
Cover page illustration is a part of a picture, showing an 8-bit micro-controller Intel 8742, which contains a single chip CPU with a speed of 12 MHz, 128 bytes of RAM, 2048 bytes EPROM and input/output.
Sameli, Ioan. 2006. Old processor. May 25th. Flickr.
http://www.flickr.com/photos/biwook/153062645/.