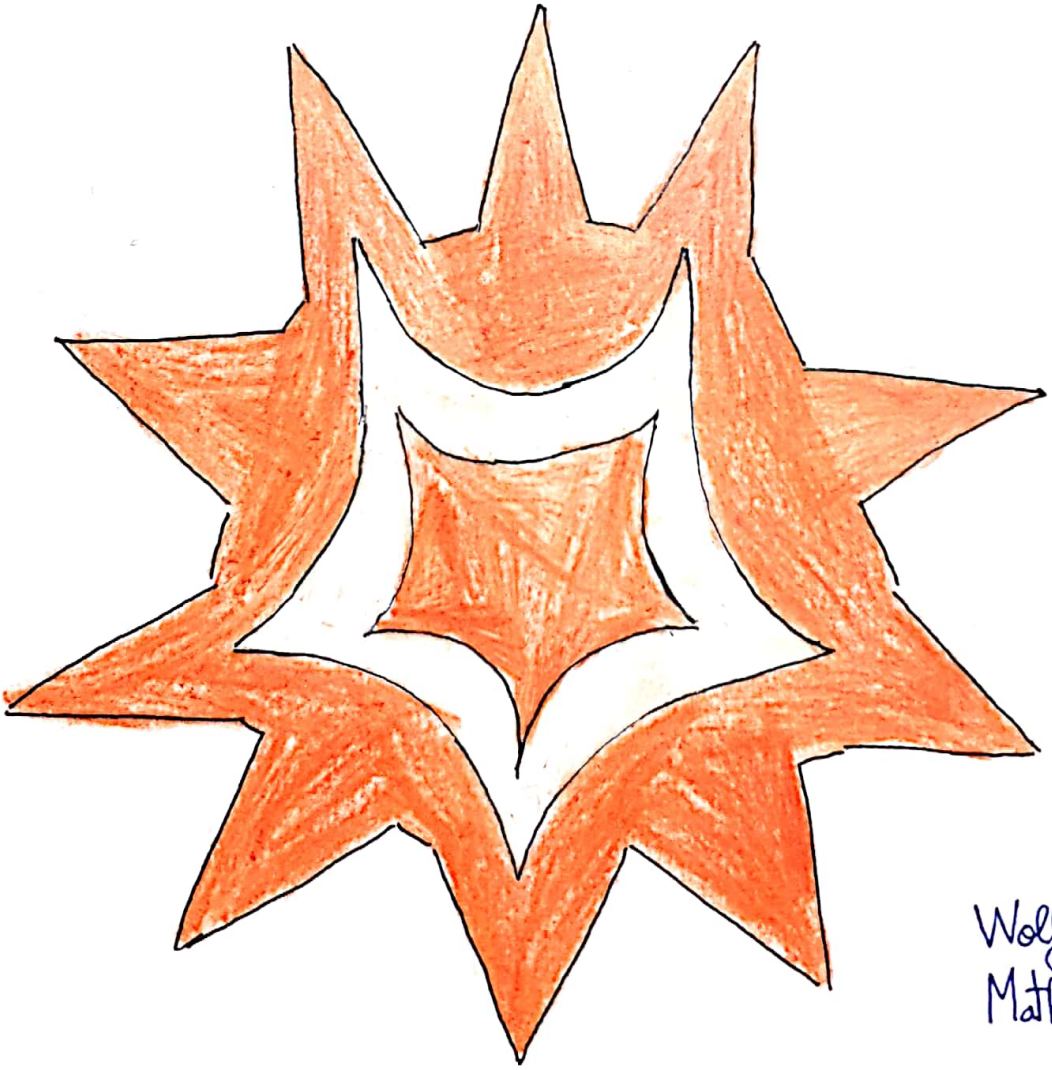
MÉTODOS



Wolfram
Mathematica

Josu P. Z.

Chapter 1

Radix conversion

We use decimal notation:

$$231'45_{10} = 2 \cdot 10^2 + 3 \cdot 10^1 + 1 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

→ We say that 231'45 is in radix ten.

Suppose we want to convert it to radix eight.

$$231'45_8 = 2 \cdot 8^2 + 3 \cdot 8^1 + 1 \cdot 8^0 + 4 \cdot 8^{-1} + 5 \cdot 8^{-2} =$$

$$= 153'578125_{10}$$

Most common ones: 10, 2, 8, 16 → hexadecimal

Radix 16:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Radix 10:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9: base of Radix 10.; 10, 11, 12, 13, 14, 15: equivalents for A, ..., F)

Radix 2:

0 0 0 0 → 0

0 0 0 1 → 1

0 0 1 0 → 2

0 0 1 1 → 3

0 1 0 0 → 4

0 1 0 1 → 5

0 1 1 0 → 6

0 1 1 1 → 7

1 0 0 0 → 8

1 0 0 1 → 9

1 0 1 0 → 10 — A

1 0 1 1 → 11 — B

1 1 0 0 → 12 — C

1 1 0 1 → 13 — D

1 1 1 0 → 14 — E

1 1 1 1 → 15 — F

Error on memory 0x ABDF8DE7! A classic error that (advises of incoming hex number) references memory location as hexadecimal number.

To binary equivalent:

$\overbrace{ABDF}\overbrace{8DE7} \longrightarrow$

$= \overbrace{10}\overbrace{1010}\overbrace{11}\overbrace{1101}\overbrace{1111}\overbrace{1000}\overbrace{1101}\overbrace{1110}\overbrace{0111}$

To octary:

$= 25367706747$

Radix 8 : (octal)

000 → 0
001 → 1
010 → 2
011 → 3
100 → 4
101 → 5
110 → 6
111 → 7

Decimal rounding:

$2'\overbrace{9}_{10} = 3_{10}$

$13'24\overbrace{7}_{8} = 13'25_8$

$101'100\overbrace{1}_{2} = 101'1010_2$

Briefing:

- From any radix to decimal: just sustitute its value ex:

$AB7_{16} = 10 \cdot 16^2 + 11 \cdot 16^1 + 7 \cdot 16^0 = 2743_{10}$

$745_8 = 7 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 485_{10}$

$1001 = 2^3 + 2^0 = 9_{10}$

- From hex to binary an vice versa, just substitute each hexadecimal character for its corresponding 4 bits (storting from left).

$AB7_{16} = \ldots 0111_2 = \underbrace{1010}_{A_{16}}\ \underbrace{1011}_{B_{16}}\ \underbrace{0111}_{7_{16}}{}_2$

$\underbrace{1}_{1}\ \underbrace{1001}_{9}\ \underbrace{0010}_{2}\ \underbrace{1011}_{B}\ \underbrace{1000}_{8_{16}} = 192B8_{16}$

(if we have at the left and incomplete group of numbers, add 0 to the left until we have a group of four bits)

This works because of powers of 2 (both bases are).

$A00_{16} = 10 \cdot 16^2 = 10 \cdot (2^4)^2 = 10 \cdot 2^8$

$\underbrace{1010}\ \underbrace{0000}\ \underbrace{0000} = 2^{12} + 2^{10} = \underbrace{(2^2+1)}_{5} \cdot 2^{10} = 10 \cdot 2^8$

- The same goes for the other relations between radix 16, 8 and 2.

Examples:

$317_8 = 011001111_2 = 207_{10}$

$01101_2 = 15_8 = 8 + 5 = 13_{10}$

(1 5)

$317_8 = 011\,00\,111 = CF_{16} = 207_{10}$

(C F)

(from radix 8 to 16, an intermediate step. Works similar the other way around) (Left to right to make the groups!)

$CF_{16} = 1100\,1111 = 011\,001\,111 = 317_8 = 207_{10}$

(3 1 7)

- From radix ten to binary. Mark all the powers of two from 2^1 to 2^x so that $2^x \geq n$. And substract the highest possible power of 2. Or, divide by two:

$109 \rightarrow$

128	64	32	16	8	4	2	1
"	"	"	"	"	"	"	"
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$109 = 64 + 45 = 64 + 32 + 13 = 64 + 32 + 8 + 4 + 1 \rightarrow$

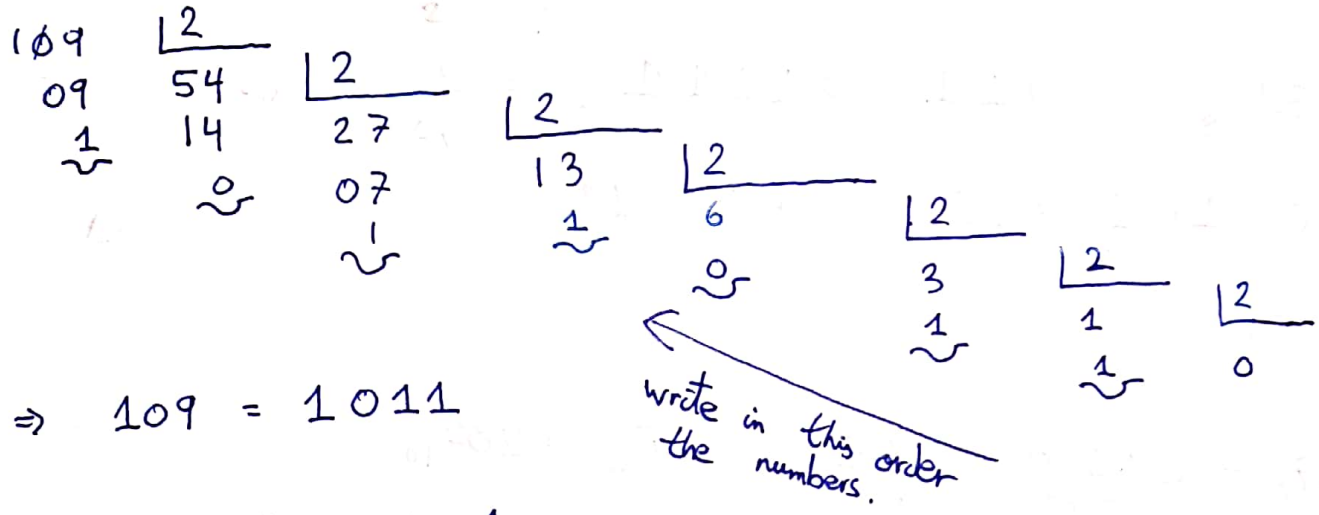
$\Rightarrow 0\,1\,1\,0\,1\,1\,0\,1 \rightarrow 109_{10} = 1101101_2 =$

$= 0110\,1101 = 001\,101\,101 \Rightarrow$

(6 D 1 5 5)

$\Rightarrow 109_{10} = 1101101_2 = 6D_{16} = 155_8$

Or:



$\Rightarrow \quad 109 = 1011$

$109 \;\%\; 2 \quad r = 1$

$54 \;\%\; 2 \quad r = 0$

$27 \;\%\; 2 \quad r = 1$

$13 \;\%\; 2 \quad r = 1$

$6 \;\%\; 2 \quad r = 0$

$3 \;\%\; 2 \quad r = 1$

$1 \;\%\; 2 \quad r = 1$

0 //

$\rightarrow \quad 109_{10} = 1101101_2$

Why does this work? Think modularly:

$109_{10} = 9 + 0 \cdot 10 + 1 \cdot 100 \Rightarrow$

$109_{10} \bmod (10) = 9 \rightarrow$ unities (know we don't mind these).

$109_{10} \bmod (100) = 09 = 0 \cdot 10 + 9$ unities $\Rightarrow$ 0 tens.

$109_{10} - (09) = 100 \bmod (1000) = 1$ hundreds.

Ex:

$238 \equiv 8 \bmod (10) \rightarrow \begin{cases} 238 \equiv 38 \bmod (100) \\ 230 \equiv 30 \bmod (100) \end{cases}$

Better seen this way:

$$109 = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \dots + a_7 \cdot 2^7 =$$

$$= a_0 + 2 \cdot (a_1 + 2 \cdot (a_2 + \dots + 2 \cdot (a_7)) \dots)$$

$\Rightarrow 109 \equiv a_0 \bmod (2) \Rightarrow a_0 =$ resto $109/2 \rightarrow 1$

$$\left(\frac{109 - a_0}{2}\right) \equiv a_1 \bmod (2) \Rightarrow a_1 = \text{resto } \left(\frac{109 - a_0}{2}\right)/2$$

Now, $\underbrace{109}_{\text{dividend}} = \underbrace{2}_{\text{divisor}} \cdot \underbrace{\left(\frac{109 - a_0}{2}\right)}_{\text{cocient}} + \underbrace{a_0}_{\text{reminder}}$

So if:

$109 = a_0 \cdot 2^0 + \dots + a_7 \cdot 2^7$ and we define

c_0 as $\frac{109 - a_0}{2}$, $c_n = \frac{c_{n-1} - a_n}{2}$ $(c_{n-1} \geq c_n)$

And thus our algorithm:

$$X = a_0 \cdot 2^0 + \dots + a_k \cdot 2^k \Rightarrow$$

$$X \equiv a_0 \bmod (2)$$

$$\frac{X - a_0}{2} = c_0 \equiv a_1 \bmod (2)$$

$$\frac{\left(\frac{X - a_0}{2}\right) - a_1}{2} \equiv \frac{c_0 - a_1}{2} \equiv a_2 \bmod (2)$$

...

We start computing $X = 2 \cdot c_0 + a_1$ then:

$\Rightarrow c_0 = 2 \cdot c_1 + a_2 = \left(\frac{X - a_1}{2}\right)$ .....

$$\begin{cases} x = 2 \cdot c_0 + a_0 \\ c_0 = 2 \cdot c_1 + a_1 \\ \vdots \\ c_k = 2 \cdot c_{k+1} + a_{k+1} \end{cases}$$

Example:

$109 = 2 \cdot 54 + 1 \rightarrow a_0 = 1$

$54 = 2 \cdot 27 + 0 \rightarrow a_1 = 0$

$27 = 2 \cdot 13 + 1 \rightarrow a_2 = 1$

$13 = 2 \cdot 6 + 1 \rightarrow a_3 = 1$ $\longrightarrow$

$6 = 2 \cdot 3 + 0 \rightarrow a_4 = 0$

$3 = 2 \cdot 1 + 1 \rightarrow a_5 = 1$

$1 = 2 \cdot 0 + 1 \rightarrow a_6 = 1$

0 /

$$\rightarrow \boxed{109_{10} = a_0 \cdot 2^0 + \dots + a_6 \cdot 2^6 + \cancel{a_7 \cdot 2^7}^{\,0} =}$$

$$= 1 \cdot 2^0 + \dots + 1 \cdot 2^6 =$$

$$= a_6 a_5 a_4 \dots a_0 (2) = \underline{1101101_2}$$

(These are my notes, now notes from class).

Our aim is to convert a b-radix number into a B-radix number.

$$u_m \dots u_2 u_1 u_0 . u_{-1} u_{-2} u_{-3} \dots u_{-n}$$

Example:

$253_8 = 2 \cdot 8^2 + 5 \cdot 8^1 + 3 \cdot 8^0 = 2 \cdot 64 + 40 + 3 =$

$= (2 \cdot 8 + 5) \cdot 8 + 3 = 3 + 8 \cdot (5 + 8 \cdot (2)) = 171_{10}$

(10 10)

What is its radix 7 equivalent?

$(2_7 \cdot 11_7 + 5_7) \cdot 11_7 + 3_7 = (22_7 + 5_7) \cdot 11_7 + 3_7 =$

$= 30_7 \cdot 11_7 + 3_7 = 330_7 + 3_7 = 337_7$

We check:

$333_7 = (3 \cdot 7 + 3) \cdot 7 + 3 = 24 \cdot 7 + 3 = 168 + 3 = 171$

Book: Donald Knuth, the art of computer programming
(.djvu format (eastern pdf *10 better compresion)).

To transform from radix-10 to other radix:

171 | 8
11 21 | 8
3 5 2 | 8
2 0

$171_{10} = 253_8$

333_7 | 11_7

03 | 30_7 | 11_7 | 11_7

3 | 05 | 2 | 0

2

$\Rightarrow 333_7 = 253_8$

(Able to change from any radix to any radix with this method!)

X_b | B_b

..... $X_b = r_n r_{n-1} \dots r_3 r_2 r_1{}_B$

For the decimal part; from any radix to radix ten, similar:

Example:

$0'132 = 1 \cdot 4^{-1} + 3 \cdot 4^{-2} + 2 \cdot 4^{-3} =$

$= \left(\left(\left(2 \cdot \frac{1}{4}\right) + 3\right) \cdot \frac{1}{4} + 1\right) \cdot \frac{1}{4} =$

$= \left(\left(\frac{1}{2} + \frac{6}{2}\right) \cdot \frac{1}{4} + 1\right) \cdot \frac{1}{4} = \left(\frac{7}{8} + \frac{8}{8}\right) \cdot \frac{1}{4} = \frac{15}{32} =$

$= 0'46875_{10}$

And the other way around: (radix 10 or other, to any radix).

$0'46875_{10} \rightarrow 0'46875 \cdot 4 = 1'875 = 1 + 0'875 \rightarrow \lfloor 1'875 \rfloor = 1$

$0'875 \cdot 4 = 3 + 0'5$ (Note, what if $23 + 0'5$?)

$0'5 \cdot 4 = 2 + 0$ //

$(0 \cdot 4 = 0 + 0)$

$\Rightarrow 0'46875_{10} = 0'132_4$

(Note: $x \in [0,1) \rightarrow 0 \leq x \cdot N \leq N \rightarrow$ we can always take the outcome.)

Remember:

Floor function → $\lfloor 3.3 \rfloor = 3$

Ceiling function → $\lceil 3.3 \rceil = 4$

Example:

$12112'\overline{02}_3 = X$

For periodic decimals.

$12112_3 = 1 \cdot 81 + 2 \cdot 27 + 1 \cdot 9 + 1 \cdot 3 + 2 \cdot 3^0 = 149$

$0'\overline{02}_3 = \frac{2}{3^2} + \frac{2}{3^4} + \frac{2}{3^6} + \dots = 2 \cdot \sum_{n=1}^{\infty} \left(\frac{1}{9}\right)^n = 2 \cdot \frac{1/3^2}{1 - 1/3^2} =$

$= \frac{\frac{2}{3^2}}{1 - 1/3^2} = \frac{1}{4}_{10}$

Example:

$X_4 = 12112'\overline{02}_3$

$X_3 = 12112'\overline{02}_3$

$100_3 \cdot X_3 = 12112\,02'\overline{02}_3$

$-(100_3 - 1_3)X_3 = 12112'\overline{02}_3 - 12112\,02'\overline{02}_3 \Rightarrow$

$\Rightarrow (21_4 - 1_4)X_4 = \begin{array}{r} 12112\,02'\overline{02}_3 \\ 12112'\overline{02}_3 \\ \hline 1122020\ / \end{array} \Rightarrow$

$\Rightarrow 20_4 X_4 = 1122020_3$

1122020_3

02

22

00

02

20

02

$4_{10} = 11_3$

$102001 \mid 11_3$

2

... (finish as exercise).

RADIX CONVERSION BRIEFING

- **Integer from base N to base 10:**

$$a_m a_{m-1} \dots a_0\ _N = a_0 \cdot N^0 + a_2 \cdot N^1 + \dots + a_m \cdot N^m = X_{10}$$

- **From base 10 to N or generally, from base N_1 to N_2:**

$$a_m a_{m-1} \dots a_0\ N_1 = b_{m'} b_{m-1'} \dots b_0\ N_1 \cdot \underbrace{(N_2)_{N_1}}_{N_2 \text{ written in base } N_1} + r_1$$

$$b_{m'} b_{m-1'} \dots b_0\ N_1 = c_{m''} b_{m-1''} \dots c_0 \cdot N_1 \cdot (N_2)_{N_1} + r_2$$

$$\vdots$$

$$Y_{m'''} Y_{m-1'''} \dots Y_0\ N_1 = \underbrace{z_{m^{IV}} z_{m-1^{IV}} \dots z_0\ N_1}_{0_{N_1}} \cdot (N_2)_{N_1} + r_n < (N_2)_{N_1}$$

$0 = //$

$$\rightarrow a_m a_{m-1} \dots a_0\ N_1 = r_n r_{n-1} \dots r_2 r_1\ N_2$$

(Note: r_i can be $0 \leq N_1 \leq r_i < N_2$, for example, when passing from radix 4 to 16 the reminder r_i can be $0 < 4 < 15 < 16$, but since $0 \leq r_i < N_2 \Rightarrow$ it's only one character in radix N_2)

(in this case, $15 = G$)

• Floating in base N to base 10:

Analogous to integer base N to base 10 but this time;

$$0.a_1a_2...a_m\ N = a_1 \cdot N^{-1} + a_2 \cdot N^{-2} + ... + a_m \cdot N^{-m} = X_{10}$$

• Floating in base N_1 to base N_2:

$0.a_1a_2...a_m\ N_1 \longrightarrow$

$0.a_1a_2...a_m\ N_1 \cdot (N_2)_{N_1} = r_1.b_1b_2...b_{m'}, \quad 0 \le r_1 < N_2$

$0.b_1b_2...b_{m'}\ N_1 \cdot (N_2)_{N_1} = r_2 \cdot c_1c_2...c_{m'}, \quad 0 \le r_2 < N_2$

$\vdots$ (* Check last note on r_i)

$0.Y_1Y_2...Y_{m'}\ N_1 \cdot (N_2)_{N_1} = r_n \cdot 0, \quad 0 \le r_n < N_2 \Rightarrow$

$\Rightarrow 0.a_1a_2...a_m\ N_1 = r_1r_2...r_n\ N_2$ //

• Pericodic base N to base 10:

$$0'b_1b_2...b_m\overbrace{a_1a_2...a_n}\ N \rightarrow \underbrace{0'b_1b_2...b_m}_{\text{previous case}}\ N + N^{-m+1} \cdot \underbrace{0'\overbrace{a_1a_2...a_n}}_{\text{what we really care}}\ N$$

$$0'\overbrace{a_0a_1...a_{n-1}}^{\text{n elements } (n-1-0+1=n)}\ N = \sum_{i=0}^{\infty} \frac{a_0}{N^{1+n\cdot i}} + ... + \sum_{i=0}^{\infty} \frac{a_{n-1}}{N^{n-1+n\cdot i}} =$$

$$= \frac{a_0}{N} \cdot \underbrace{\sum_{i=0}^{\infty} \left(\frac{1}{N^n}\right)^i}_{\text{formula}} + ... + \frac{a_{n-1}}{N^{n-1}} \cdot \underbrace{\sum_{i=0}^{\infty} \left(\frac{1}{N^n}\right)^i}_{\text{formula}} =$$

$$= \frac{\overbrace{\left(\frac{1}{N^n}\right)^{k_0=0}}^{1}}{1-\frac{1}{N^n}} \cdot \left[\frac{a_0}{N} + ... + \frac{a_{n-1}}{N^{n-1}}\right] = X_{10}$$

- Periodic N_1 to N_2 (full number):

$$a_1 \dots a_m . a_1' \dots a_{m'} \overline{b_1 \dots b_n}_{N_1} = X_{N_2} \rightarrow ?$$

$$\Rightarrow (10^{n+m'}_{N_1}) \cdot a_1 \dots a_m . a_1' \dots a_{m'} \overline{b_1 \dots b_n} = a_1 \dots a_m a_1' \dots a_{m'} b_1 \dots b_n . \overline{b_1 \dots b_n}_{N_1}$$

$$(10^{m'}_{N_1}) \cdot a_1 \dots a_m . a_1' \dots a_{m'} \overline{b_1 \dots b_n} = a_1 \dots a_m a_1' \dots a_{m'} . \overline{b_1 \dots b_n}_{N_1} \Rightarrow$$

$$(10^{n+m'} - 10^{m'})_{N_1} \cdot \underbrace{a_1 \dots a_m . a_1' \dots a_{m'} \overline{b_1 \dots b_n}}_{X_{N_2}} = \left[\begin{matrix} a_1 \dots a_m a_1' \dots a_{m'} b_1 \dots b_n - \\ - a_1 \dots a_m a_1' \dots a_{m'} \end{matrix} \right]$$

$$\Rightarrow a_1 \dots a_m . a_1' \dots a_{m'} \overline{b_1 \dots b_n}_{N_1} = \frac{\left[a_1 \dots a_m a_1' \dots a_{m'} b_1 \dots b_n - a_1 \dots a_m a_1' \dots a_{m'} \right]_{N_1}}{(10^{n+m'} - 10^{m'})_{N_1}}$$

(Transform this very last fraction) $\Rightarrow$

$$X_{N_2} = \frac{\left(\left[a_1 \dots a_m a_1' \dots a_{m'} b_1 \dots b_n - a_1 \dots a_m a_1' \dots a_{m'} \right] \right)_{N_2}}{\left(\left[10^{n+m'} - 10^{m'} \right] \right)_{N_2}}$$

- X_{N_2} is not necessarily the irreducible fraction, to achieve this on intermediate step from $N_1 \rightarrow$ base 10 $\rightarrow$ reduce $\rightarrow N_2$ might be necessary.
- They might ask for the decimal expansion, in which case we have to do the division in base N_2 of the respective fraction.

Note: Don't mix any of these methods.
Each one of them has its own demonstration, so using the integer methods to the floating decimals its a big mistake!

Example:

$x_3 = 12112.\overline{02}_3 \xrightarrow{\text{to}}$ radix 4?

$100_3 \cdot x_3 = 1211202.\overline{02}_3$

$(100_3 - 1_3)x_3 = (21_4 - 1_4)x_4 = 20_4 x_4 = 1211202.\overline{02}_3 - 12112.\overline{02}_3 = 1122020_3$

1122020_3 | 11_3 (4)

02 → 102001

22 → 100

000 → 10

02 → 101

20 → 2

02

11_3 | 2202 → 002

11_3 | 200 → 20 → 2

11 | 11 → 0

11 | 1 → 1

11 | 0 //

$\Rightarrow 1122020_3 = 102222_4$

Another method:

$((((1\cdot 3+1)\cdot 3+2)\cdot 3+2)\cdot 3+2)\cdot 3+2)$

10_4

32	230	2012
3	3	3
12	0	12
21	21	03
222	12	02
	2010	12
		1210_2

(Incorrect result)

Solution:

$((((((1\cdot 3+1)\cdot 3+2)\cdot 3+2)\cdot 3+0)\cdot 3+2)\cdot 3+0)$

10_4

32_4

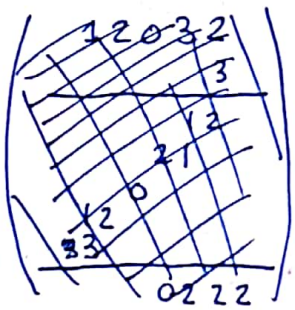
230_4

2010_4

$12030_4 + 2_4$

102222_4

2010
3
30
3
120
12030



12032
3
12
21
00
12
3
102222

10222_4 | 24 → 21111 | 104 = 2111.1

02
02
02
0

$\Rightarrow$ $12112.0\widehat{2}_3 = 2111.1_4$ //

Example:

$2111.1_4 \longrightarrow$ Radix 16 ?

Remember: $16 = 4^2 \rightarrow$ groups of 2

$\underbrace{21}_{9_{10}}\ \underbrace{11}_{5_{10}}.\ \underbrace{10}_{4_{10}}{}_4 = 0 \times 95.4 = 95.4_{16}$ (h, H)

$9_{10} = 9$, $5_{10} = 5$, $4_{10} = 4$

Computer Arithmetic

How do computers store numbers? The IEEE754 standard.

Storing numbers in 64 bits.

sign	exponent	number
(1 bit)	(11 bits)	(52 bits)
sign	c	f

$$(-1)^s \cdot 2^{c-1023} \cdot (1 + f)$$

(s = 0 or 1)

c: exponent

-1023: offset

f → mantisa

Example:

27.56640625

- Since it's positive $\Rightarrow (-1)^s = 1$
- There exist: $\exists n \mid 2^n \leq x < 2^{n+1} \Rightarrow \frac{x}{2^n} = 1.\ldots$ We choose that power of 2: 2^n ($1023+n$, because of the offset).
- We add $\{27.56640625 / 2^n\} = f$ (the mantisa).

So:

$$\begin{cases} s = 0 \\ c = 1027 \\ f = 0.722900390625 \end{cases}$$

Note: $0_{10} = 0_2$, $1_{10} = 1_{10} \Rightarrow$ s is already binary, no operations needed

$$\Rightarrow 27.56640625 = (-1)^0 \cdot 2^{1027-1023} \cdot (1 + 0.722900390625)$$

Now we pass these numbers to binary. (s is already binary)

- $1027_{10} = 1024 + 2 + 1 = 2^0 + 2^1 + 2^{10} = 10000000011_2$

- $u_{-1} = \lfloor 2 \cdot 0.722900390625 \rfloor = 1 \quad (1.445800078125)$

$1.445800078125 \cdot 2 = 0.8916015625 \rightarrow u_{-2} = 0$

$u_{-3} = \lfloor 2 \cdot 0.8916015625 \rfloor = \lfloor 1.78320325 \rfloor = 1$

$u_{-4} = \lfloor 2 \cdot 0.78320325 \rfloor = \lfloor 1.56640625 \rfloor = 1$

$u_{-5} = \lfloor 2 \cdot 0.56640625 \rfloor = \lfloor 1.1328125 \rfloor = 1$

$u_{-6} = \lfloor 2 \cdot 0.1328125 \rfloor = \lfloor 0.265625 \rfloor = 0$

$u_{-7} = \lfloor 2 \cdot 0.265625 \rfloor = \lfloor 0.53125 \rfloor = 0$

$u_{-8} = \lfloor 2 \cdot 0.53125 \rfloor = \lfloor 1.0625 \rfloor = 1$

$u_{-9} = \lfloor 2 \cdot 0.0625 \rfloor = \lfloor 0.1250 \rfloor = 0$

$u_{-10} = \lfloor 2 \cdot 0.125 \rfloor = \lfloor 0.25 \rfloor = 0$

$u_{-11} = \lfloor 2 \cdot 0.25 \rfloor = \lfloor 0.5 \rfloor = 0$

$u_{-12} = \lfloor 2 \cdot 0.5 \rfloor = \lfloor 1 \rfloor = 1$

$u_{-13} = \lfloor 2 \cdot 0 \rfloor$ // we finish here.

$0.722900390625_{10} = 1011100100010000_2 = f$

So the binary representation of our number is:

$0 \mid 10000000011 \mid 1011100100010000000\ldots0 \Rightarrow$

s — c — f — the remaining bits up to 52 spaces are 0.

$27.56640625_{10} =$

$= 0\,10000000011\,1011100100010000\ldots0$ in IEEE754

Why the offset? $0, \ldots, 2^{11}-1 = 2047$ possible exponents. To represent exponents from $-1023, \ldots, 1024$, we put the offset -1023, and so the stored c remains positive while the exponent can be both positive and negative.

$\frac{2^{11}}{2} - 1 = 1023$ (possible positive and negative exponents, 2047 total).

Instead of 64, we could have other possible representations.

16 bits →	5	bits for the exponent	
32 bits →	8	" " "	" } Most important and
64 bits →	11	" " "	" } used ones.
128 bits →	15	" " "	"
256 bits →	19	" " "	"

Example:

40 bits machine (1 bit sign, 8 exponent, 31 mantissa)

Conver to it's floating representation: 0x4B923A0C80

To binary (four bits by four)

0 (number is positive) (offset = $\frac{2^8}{2} - 1 = 127$)

0 | 1001011 | 1001 0010 0011 1010 0000 1011 1000 0000$_2$

4 B 9 2 3 A 0 C 8 0

sign — exponent — mantissa

$s = 0$

$c = 10010111_2 = 1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^4 + 1\cdot 2^7 = 151$

$f = 00100100011101000001011100000000 =$

$= 2^7 + 2^8 + 2^9 + 2^{11} + 2^{17} + 2^{19} + 2^{20} + 2^{21} + 2^{25} + 2^{28} =$

$= 1 + 2^{-3} + 2^{-6} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-13} + 2^{-19} + 2^{-20} + 2^{-23}$

$= 1 + 2^{-3} + 2^{-6} + 2^{-9} + 2^{-10} + 2^{-11} + 2^{-}$ check blackboard

$= 2^{-3} + 2^{-6} + 2^{-10} + 2^{-11} + 2^{-12} + 2^{-14} + 2^{-20} + 2^{-21} + 2^{-24}$

$X = (-1)^0 \cdot 2^{151-127} \cdot f \xrightarrow[2^{24}]{\text{distribute}}$

$X = (2^{24} + 2^{21} + 2^{18} + 2^{14} + 2^{13} + 2^{12} + 2^{10} + 2^4 + 2^3 + 2^0) =$

$= 19166233_{10}$

$\Rightarrow$ 0x4B923A0C80 = 19166233_{10}

(40 bits representation)

How could we represent ∞? And NaN? What is the largest possible number? The smallest? Periodic numbers? Round numbers?
64 bit repr.
Wikipedia of IEEE754

Decimal Machine Numbers

Normalized decimal point representation:

$$\pm 0.d_1 d_2 \dots d_k \times 10^n, \quad 1 \le d_1 \le 9 \text{ and } 0 \le d_i \le 9$$

Suppose $y \in \mathbb{R}$. We denote the floating point representation, the number the machine really stores as; $fl(y)$.
Suppose the mantisa can only store k digits and y has more than k digits. There are two strategies:

- Chopping / Truncating:
  Just discard the extra digits.

$$y = 0.d_1 d_2 \dots d_k d_{k+1} d_{k+2} \dots \Rightarrow fl(y) = 0.d_1 d_2 \dots d_k$$

- Rounding:
  Round the number. (Important for accuracy and errors).

$$y = 0.d_1 d_2 \dots d_k d_{k+1} d_{k+2} \dots 10^n$$

$$fl(y) = 0.d_1 d_2 \dots d_k + \delta \cdot 5 \cdot 10^{n-(k+1)} \quad \text{with}$$

$$\delta = \begin{cases} 1 & \text{if } d_{k+1} \ge 5 \\ 0 & \text{if } d_{k+1} < 5 \end{cases} \qquad (d_1 \in [1,9],\ d_i \in [0,9])$$

(in case $\delta = 1$, $d_i = d_i + 1$ until i_0 where $d_{i_0} + 1 \le 9$)

We explain further the IEEE754 standar:

- Smallest possible number that can be represented:
  (0s for every bit)(No, $0 \leq |x|$, we want smallest $|x|$)

$$\begin{cases} c = 0 \\ f = 0 \end{cases}$$

$$\longrightarrow 1 \cdot 2^{0-1023} \cdot (1+0) = \boxed{2^{-1023}}$$

- Number closest to 1 that can be represented: $(x \neq 1)$

$$(-1)^0 \cdot 2^{1023-1023} \cdot \left(1 + \frac{1}{2^{52}}\right) = \boxed{1 + 2^{-52}} \quad \text{and}$$

we call:

- Epsilon of the machine; $eps = 2^{-52} \simeq 2.22 \cdot 10^{-16}$
- Greatest number that can be represented:
  (1s for every bit)(but positive number $\Rightarrow s = 0$)

$$\sum_{i=0}^{n} 2^{-i}$$

$$\begin{cases} s = 1 \\ c = 2047 \\ f = 2^{-1} + 2^{-2} + \ldots + 2^{-52} \end{cases} \longrightarrow (-1)^0 \cdot 2^{2047-1023} \cdot \left(1 + \sum_{i=1}^{n} 2^{-i}\right) =$$

$$\overset{(1)}{=} 2^{1023} \cdot 2 \cdot \left(1 - 2^{-53}\right) = \boxed{2^{1023} \cdot \left(2 - 2^{-52}\right)}$$

In (1) we use:

$$\left.\begin{aligned} S_n &= r^0 + \ldots + r^n \\ S_n \cdot r &= r^1 + \ldots + r^n + r^{n+1} \end{aligned}\right\} S_n(1-r) = r^0 - r^{n+1} \Rightarrow r = 1/2$$

$$\Rightarrow S_n = \sum_{i=0}^{n} 2^{-i} = \frac{1 - 2^{-53}}{1 - 1/2} = 2 \cdot \left(1 - 2^{-53}\right) = \left(2 - 2^{-52}\right)$$

$1/2$

Representation of certain numbers:

One: 1

$$\begin{cases} s = 0 \\ c = 1023 \\ f = 0 \end{cases} \rightarrow (-1)^0 \cdot 2^{1023-1023} \cdot (1+0) = 1 \quad //$$

Zero: 0

$$\begin{cases} s = \{0, 1 \\ c = 0 \\ f = 0 \end{cases} \rightarrow \pm 2^{0-1023} \cdot (1+0) = \pm 2^{-1023} \equiv \pm 0 \quad //$$

(All 0s in the 64 bit representation).

Infinity: $\pm\infty$

$$\begin{cases} s = \{0, 1 \\ c = 2046 \\ f = 0 \end{cases} \rightarrow \pm 2^{2046-1023} \cdot (1+0) = \pm 2^{1023} \equiv \pm\infty \quad //$$

NaN: (Non Arithmetic Numbers; Not a Number; wrong number)

Examples of NaN: $\sin(\infty)$, $\cos(\infty)$, $0/0$, ∞/∞, ...

$$\begin{cases} s = \{0, 1 \\ c = 2046 \\ f \neq 0 \end{cases} \rightarrow \pm 2^{2046-1023} (1+f) = \pm 2^{1023} (1+f) \equiv NaN \quad //$$

Denormalized numbers: (Denormal numbers)

$$\begin{cases} s = \{0, 1 \\ c = 0 \\ f \neq 0 \end{cases} \rightarrow \pm 2^{-1023} (1+f) \rightarrow 0$$

Numbers greater than 0, and more important, smaller than 2^{-1022}

I.E:

$\underline{0}\,|\,\underbrace{00000000000}_{11}\,|\,\underbrace{11000\ldots.}_{52}\,0$

Instead of having $(1+f)$, this numbers have only (f) so that they can be smaller; (also, they are 2^{-1022}).

$$2^{-1023}\left(2^{-1}+2^{-2}\right) = 0'75 \cdot 2^{-1022}$$

Here is the explanation:

1) $0\,|\,00000000001\,|\,00\ldots.$

$$2^{1-1023} = 2^{-1022}$$

2) $0\,|\,00000000000\,|\,00\ldots.$

$$2^{0-1023} \equiv 0$$

To represent $2^{-1023} < X < 2^{-1022}$, we adopt the system (which is versatile and use the following representation).

3) $0\,|\,00000000000\,|$ * not zero *

$$2^{-1022} \cdot (f)$$

Note this is only if $f \neq 0$, if $f = 0$, then $X = 2^{-1023} \equiv 0$

Smallest possible denormd number:

$$\begin{cases} S = \begin{cases} 0 \\ 1 \end{cases} \\ C = 0 \\ f = 0\ldots.1 \end{cases} \rightarrow 2^{-1022} \cdot 2^{-52} \;//$$

Largest possible denormal number:

$$\begin{cases} S = \begin{cases} 0 \\ 1 \end{cases} \\ C = 0 \\ f = 11\ldots1 \end{cases} \rightarrow \pm 2^{-1022} \cdot \underbrace{\left(\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{52}}\right)}_{\text{(explanation page 21)}} = \pm$$

$$= \pm 2^{-1022} \cdot \left(1 - \frac{1}{2^{52}}\right)$$

Rounding

- Rounding towards nearest, ties to even:
  - $0.\underbrace{1101}_{\text{mantissa}}01 \longrightarrow 0.1101$
  - $0.1101\underline{11} \longrightarrow 0.111$
  - $0.1101\underline{1} \longrightarrow \begin{cases} 0.1101 \\ 0.1110 \leftarrow \text{default option} \end{cases}$

(Both options are equally close, we choose the significant last bit to be 0, we tie to even).

- Rounding towards 0
- Rounding towards $+\infty$
- Rounding towards $-\infty$
- Recent option:
  It is used for decimal, not to binary. It consists of rounding to nearest, ties away from zero. (Example, 4 decimal places).

  - $0.\overline{23474} \longrightarrow 0.2374$
  - $0.23476 \longrightarrow 0.2378$
  - $0.23475 \longrightarrow 0.2378$

} if last $\ell \in \{0,\ldots,4\}$, we truncate;
if last $\ell \in \{5,\ldots,9\}$, we add +1 to previous.

Remember: (generally, two options).

→ chopping or truncating

→ rounding

Now, consider a five-digit floating point representation with chopping.

$$fl(1.23578) = fl(0.123578 \cdot 10^1) =$$

$$= 0.12357 \cdot 10^1 = 1.2357$$

Consider the same number, but rounding instead of chopping.

$$fl(1.23578) = fl(0.123578 \cdot 10^1) =$$

$$= 0.12358 \cdot 10^1 = 1.2358$$

Def.:

Let p^* be an approximation of $p \in \mathbb{R}$. The absolute error is defined as:

$$\text{Abs. error.}: |p - p^*|$$

And the relative error:

$$\text{Relative. error.}: \frac{|p - p^*|}{|p|}, \quad p \neq 0$$

Def.:

p^* approximates to p to t significant digits if and only if (if and only if = iff) t is the largest nonnegative number such that (such that = s.t.).

$$\frac{|p - p^*|}{|p|} \leq 5 \cdot 10^{-t}$$

(Such that the rel. error $\leq 5 \cdot 10^{-t}$).

$0'01 \leq 0'05 = 5 \cdot 10^{-2}$
$\rightarrow$ 2 digits

$0'06 > 0'05$
$0'06 \leq 0'5 = 5 \cdot 10^{-1}$
$\rightarrow$ 1 digit
$0'05 \leq 5 \cdot 10^{-2} \rightarrow$ 2 digit

Example:
Suppose we consider k-digit chopping arithmetic. Error?

$$y = 0.d_1 d_2 \dots d_k d_{k+1} \dots \cdot 10^n \quad (d_1 \neq 0)$$

Then

$$fl(y) = 0.d_1 d_2 \dots d_k \cdot 10^n = y^*$$

The error:

$$\frac{|y - fl(y)|}{|y|} = \frac{|0.0\dots0 d_{k+1} \dots \cdot 10^n|}{|y|} = \frac{|0.d_{k+1} \dots \cdot 10^{n-k}|}{|0.d_1 \dots d_k \dots \cdot 10^n|}$$

$$= \frac{|0.d_{k+1} d_{k+2} \dots|}{|0.d_1 d_2 \dots d_k d_{k+1} \dots|} \cdot 10^{-k} \leq \frac{1}{0.1} \cdot 10^{-k} = 10^{1-k} <$$

$< 5 \cdot 10^{1-k} \Rightarrow k-1$ significant digits.

Note:
$0.d_{k+1} d_{k+2} \dots \leq 1$
Since $d_1 \neq 0 \Rightarrow 0.d_1 d_2 \dots d_k d_{k+1} \dots \geq 0.1 \Rightarrow \frac{1}{0.d_1 \dots d_k \dots} \leq \frac{1}{0.1}$

Homework:

- Compute the number of significant digits when using k-digit rounding arithmetic. (Note, it depends if $0 \leq d_{k+1} \leq 4$ or $5 \leq d_{k+1} \leq 9$).

In general, when using a computer, all the operations are rounded, chopped, etc. So truly:

$$a + b + c \xrightarrow{\text{computer}} fl(a) + fl(b) + fl(c)$$

- Because of floating point approximation and so, associativity may not hold true for lots of operation.

In math:

$$(a + b) + c = a + (b + c)$$

But in computer science:

- $(a+b)+c = fl(fl(fl(a) + fl(b)) + fl(c))$

$$a + (b+c) = fl(fl(a) + fl(fl(b) + fl(c)))$$

And for certain cases it doesn't hold true (for most of them because of approximation.

Example next page.

Example:

$$ax^2 + bx + c = 0$$

$$a\left(x^2 + \frac{b}{a}x + \frac{c}{a}\right) = 0$$

$$a\left(\left(x + \frac{b}{2a}\right)^2 - \frac{b^2}{4a^2} + \frac{c}{a}\right) = 0$$

$$\left(x + \frac{b}{2a}\right)^2 = \frac{b^2}{4a^2} - \frac{c}{a} = \frac{b^2 - 4ac}{4a^2}$$

$$\boxed{x = -\frac{b}{2a} \pm \sqrt{\frac{b^2 - 4ac}{2a}} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

$$= \frac{(-b \pm \sqrt{b^2 - 4ac})}{2a} \cdot \frac{(-b \mp \sqrt{b^2 - 4ac})}{(-b \mp \sqrt{b^2 - 4ac})} = \frac{(-b)^2 - (\sqrt{b^2 - 4ac})^2}{2a(-b \mp \sqrt{b^2 - 4ac})} =$$

$$= \frac{4ac}{2a(-b \mp \sqrt{b^2 - 4ac})} = \boxed{\frac{2c}{-b \mp \sqrt{b^2 - 4ac}}}$$

Example:

Let us consider: $x^2 + 62.10x + 1 = 0$ (with calculator)

$x_1 = -0'016107237 41$

$x_2 = -62'08389276$

Let's approximate x_1 and x_2 by using 4-digit rounding

In general try to avoid sums of addens with different signs and very different sizes.

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{}$$

1) $fl\left(\sqrt{b^2-4ac}\right)$

$62.10^2 = 3856.41 = 0.385641 \cdot 10^4 \Rightarrow$

$\Rightarrow fl(62.10^2) = 3856$

2)

$fl\left(\sqrt{b^2-4ac}\right) = fl\left(\sqrt{fl(62.10^2) - 4}\right) =$

$= fl(\sqrt{3852}) = fl(62.0644) = 62.06$

So:

- $x_1 = fl\left(\frac{fl(-62.10 + 62.06)}{2}\right) = fl\left(\frac{fl(-0.04)}{2}\right) =$

$= fl(-0.02) = -0.02$

- $x_2 = fl\left(\frac{fl(-62.10 - 62.06)}{2}\right) = fl\left(\frac{fl(-124.16)}{2}\right) =$

$= fl\left(\frac{-124.2}{2}\right) = fl(-62.1) = -62.1$

Let's compare the relative errors.

$$\begin{cases} x_1 = -0.01610723 \; ; \; fl(x_1) = -0.02 \\ x_2 = -62.08390 \; ; \; fl(x_2) = -62.1 \end{cases}$$

$$\frac{|x_1 - fl(x_1)|}{|x_1|} \approx 2.4 \cdot 10^{-1} \leq 5 \cdot 10^{-1} \rightarrow 1 \text{ significant digit (1 its correct).}$$

$$\frac{|x_2 - fl(x_2)|}{|x_2|} \approx 3.2 \cdot 10^{-4} \leq 5 \cdot 10^{-4} \rightarrow 4 \text{ significant digits (4 correct digits).}$$

This can be justified because we had to substract for x_1 which is bad for precission.

Let's use the other formula for x_1:

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}} \Rightarrow fl(x_1) = fl\left(\frac{2}{fl(-62.1 - \underbrace{62.06}_{\text{from before}})}\right) =$$

$$= fl\left(\frac{2}{fl(-124.16)}\right) = fl\left(\frac{2}{-124.2}\right) = -0.0161$$

$$\Rightarrow \frac{|x_1 - fl(x_1)|}{|x_1|} \approx 6.2 \cdot 10^{-4} \leq 5 \cdot 10^{-3} \rightarrow 3 \text{ significant digits (3 correct).}$$

So there's a difference, a noticeable one. Note that for the second case, despite having 4 numbers equal, our system identifies three equal, that's because we defined it this way.

(Note; for decimal machines: $(-1)^s \cdot 10^x \cdot (1+f)$, but the rest remains equal.)

Fixed point representation:

We reserve certain bits for the integer and other for the mantissa.

The habitual thing is a floating point representation (as we have already studied for IEEE 754).

Exercise: solution.

Let $y = 0.d_1 \dots d_k d_{k+1} \dots \cdot 10^n$ and $fl(y)$ the digit (k-digit) approximation using rounding arithmetic. Obtain f significant digits.

$$fl(y) = \begin{cases} 0.d_1 \dots d_k \cdot 10^n & \text{if } d_{k+1} < 5 \\ (0.d_1 \dots d_k + 1 \cdot 10^{-k}) \cdot 10^n & \text{if } d_{k+1} \geq 5 \end{cases}$$

($d_1 \neq 0$ because we normalize the number.)

If $d_{k+1} < 5$:

$$\left| \frac{y - fl(y)}{y} \right| = \left| \frac{0.d_1 d_2 \dots d_k d_{k+1} \dots - 0.d_1 d_2 \dots d_k}{0.d_1 d_2 \dots d_k d_{k+1} \dots} \right| \cdot \frac{10^n}{10^n} = \left| \frac{0.d_{k+1} \dots \cdot 10^{-k}}{0.d_1 \dots d_{k+1} \dots} \right| \overset{(1)}{\leq} \frac{0.5}{0.1} \cdot 10^{-k} = 5 \cdot 10^{-k}$$

$\Rightarrow$ k digits of precision.

(Because $d_{k+1} < 5$ and $d_1 \neq 0$)

if $d_{k+1} \geq 5$

$$\left|\frac{y - fl(y)}{y}\right| = \left|\frac{0.d_1 d_2 \ldots d_k d_{k+1} \ldots - (0.d_1 \ldots d_k + 10^{-k})}{0.d_1 d_2 \ldots d_k d_{k+1} \ldots}\right| \cdot \frac{10^n}{10^n}$$

$$= \left|\frac{0.d_{k+1} \ldots \cdot 10^{-k} - 10^{-k}}{0.d_1 \ldots d_k \ldots}\right| = \left|\frac{0.d_{k+1} \ldots - 1}{0.d_1 \ldots d_k \ldots}\right| \cdot 10^{-k} \leq$$

($d_{k+1} \geq 5$)

$$\leq \frac{|0.5 - 1|}{0.1} \cdot 10^{-k} = 5 \cdot 10^{-k} \Rightarrow \boxed{k \text{ correct digits.}}$$

Example:

$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$, $x = 4.71$

Consider 3 digit rounding arithmetic: ($f(x) = -14.263899$)

$f(x) = x^2(x - 6.1) + 3.2x + 1.5 =$ (They could give us or we would have to compute it with the calculator.)

$= x(3.2 + x(x - 6.1)) + 1.5 =$

$= ((x - 6.1)x + 3.2)x + 1.5$ (→ Nested manner).

$a = fl(x - 6.1) = fl(4.71 - 6.1) = fl(-1.39) = -1.39$

$b = fl(a \cdot x) = fl(-1.39 \cdot 4.71) = fl(-6.5469) = -6.55$

$c = fl(b + 3.2) = fl(-6.55 + 3.2) = fl(-3.35) = -3.35$

$d = fl(c \cdot x) = fl(-3.35 \cdot 4.71) = fl(-15.7785) = -15.8$

$e = fl(d + 1.5) = fl(-15.8 + 1.5) = fl(-14.3) = -14.3$

→ $\boxed{f(4.71) = -14.3}$

Use calculator for the middle steps.

Relative error and number of significant digits:

$$\frac{|y - fl(y)|}{|y|} = \frac{|-14.263899 + 14.3|}{|-14.263899|} = \boxed{2'5 \cdot 10^{-3}} \leq 5 \cdot 10^{-3}$$

2'5 · 10^-3 : relative error

$\Rightarrow$ **3 significant digits.**

This method is much better than doing the operations from left to right, we perform less operations and it's more precise. (With normal method → f(x) = -13. something and 1 significant digit).

If they don't give y, use the calculator to get it.

- Usual method:

$a = fl(x) = 4.71$

$b = fl(x \cdot fl(x)) = fl(4.71 \cdot 4.71) = fl(22.1841) = 22.2$

$c = fl(x \cdot fl(x \cdot fl(x))) = fl(4.71 \cdot 22.2) = fl(104.562) = 105$

$d = fl(-6.1 \cdot b) = fl(-6.1 \cdot 22.2) = fl(-135.42) = -135$

$e = fl(c - d) = fl(105 - 135) = -30$

$f = fl(3.2 \cdot a) = fl(4.71 \cdot 3.2) = fl(15.072) = 15.1$

$g = fl(e + f) = fl(-30 + 15.1) = fl(-14.9) = -14.9$

$\boxed{f(x) = fl(g + 1.5) = fl(-14.9 + 1.5) = fl(-13.4) = -13.4}$

Relative error:

(next page)

$$\left|\frac{y - fl(y)}{y}\right| = \left|\frac{-14.263899 + 13.4}{-14.263899}\right| = \boxed{6'05654 \cdot 10^{-2} \leq 5 \cdot 10^{-1}}$$

$\Rightarrow$ **1 correct digit.**

Consider 3 digit Chopping arithmetic: (Nested manner)

$$f(x) = ((x - 6.1) \cdot x + 3.2) \cdot x + 1.5$$

$a = fl(x - 6.1) = fl(-1.39) = -1.39$

$b = fl(a \cdot x) = fl(-6.5469) = -6.54$

$c = fl(b + 3.2) = fl(-3.34) = -3.34$

$d = fl(c \cdot x) = fl(-15.7314) = -15.7$

$e = fl(d + 1.5) = fl(-14.2) = -14.2$

$$\boxed{f(4.71) = -14.2}$$

$$\left|\frac{y - fl(y)}{y}\right| = \left|\frac{-14.263899 + 14.2}{-14.263899}\right| = \boxed{4.479 \cdot 10^{-3} \leq 5 \cdot 10^{-3}}$$

$\Rightarrow$ **3 significant digits.**

Here it wasn't the case, but usually chopping is less precise than rounding, as we proved before.

Let's compute some matrix algorithms now.