Finite representation of real numbers

Fixed-point numbers

Dr. Ing. Rodrigo Gonzalez

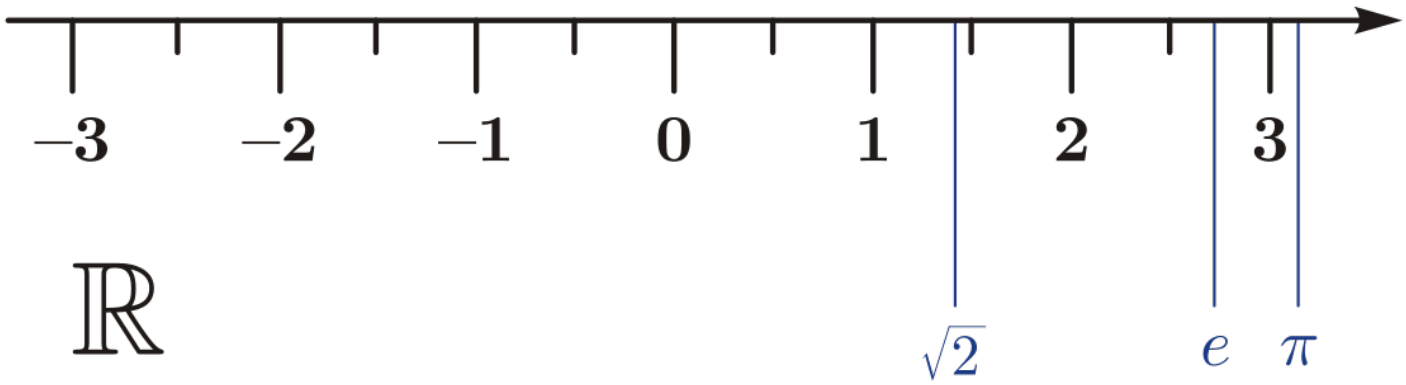
rodralez@frm.utn.edu.ar
rodrigo.gonzalez@ingenieria.uncuyo.edu.ar

UTN FACULTAD REGIONAL MENDOZA
UNIVERSIDAD TECNOLÓGICA NACIONAL

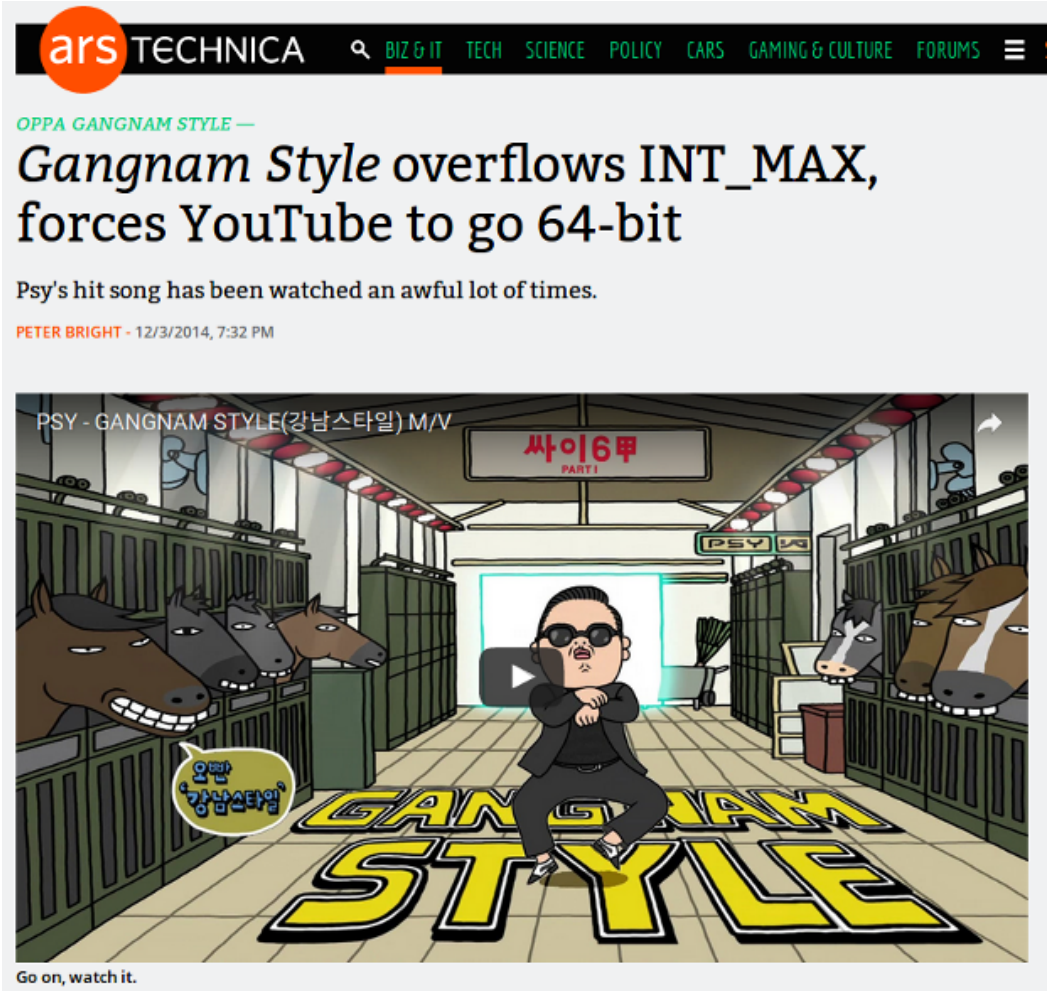
UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Summary

- It is impossible to represent infinite numbers in computers.
- Programmers must choose the best real number representation for a particular application.



Todas las representaciones tienen ventajas y desventajas.
Ej: Punto fijo, punto flotante, entero



https://arstechnica.com

Patriot derriba un misil con otro misil

- The radar of a Patriot missile system is designed to detect an incoming missile twice in order to avoid false alarms.
- Time is stored to an accuracy of 1/10th of a second in a 24-bit register.
- It results in $0.0001110110011001100110011001101\ldots$ with an infinite number of bits.
- The error of representing 1/10th in 24-bit register is 0.000000095 decimal of seconds.
- After 100 hours of operation, cumulative error gives $0.000000095 \times 100 \times 60 \times 60 \times 10 = 0.34$ s.
- A SCUD travels at about 1,676 m/s. In 0.34 s, it travels about 600 meters.
- This error in the time calculation caused the Patriot system to expect an incoming missile at a wrong location for the second detection, causing it to consider the first detection as false alarm.
- On February 25th, 1991, a Patriot Missile system at Dhahran, Saudi Arabia failed to intercept a SCUD missile, killing 28 American soldiers.

More information at https://blog.penjee.com/famous-number-computing-errors/

Unsigned integers:

- An N-bit binary word can represent a total of 2^N separate values.
- Range: 0 to $2^N - 1$
- $n_{10} = 2^{N-1}b_{N-1} + 2^{N-2}b_{N-2} + \cdots + 2^1 b_1 + 2^0 b_0$

Bit Pattern	Unsigned	2's Complement
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
•	•	•
•	•	•
0111 1110	126	126
0111 1111	127	127
1000 0000	128	-128
1000 0001	129	-127
•	•	•
•	•	•
1111 1110	254	-2
1111 1111	255	-1

Two's complement signed integers:

- Range: -2^{N-1} to $2^{N-1} - 1$.
- $n_{10} = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

Complemento a 2 permite eficiencia de hardware,
Usa un sumador tanto para sumar como para restar

in C:

Como resto bN-1 los que arranquen en 1 son negativos

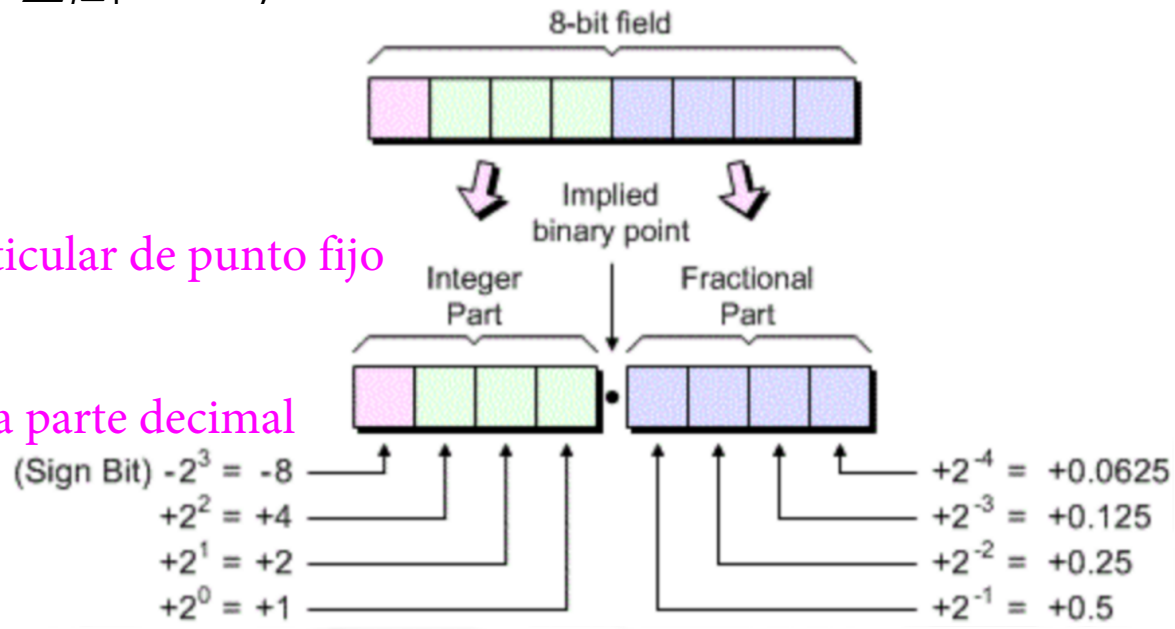
- 8 bits (`char, int8_t`): [-128, 127]]
- 16 bits (`short, int16_t`): [-32768, 32767]
- 32 bits (`int, long, int32_t`): [-2147483648, 2147483647]

char,short,int,long, puede cambiar segun compilador. Los otros son fijos

In fixed-point representation, a real number x is represented by an integer X with $N = m + n + 1$ bits, where:

- N is the wordlength.
- m represents the number of integer bits (to the left of the binary point).
- n represents the number of fractional bits (to the right of the binary point).
- One extra bit for the sign bit.
- The weights of bits to the right of the binary point are negative powers of 2: $2^{-1} = \frac{1}{2}$, $2^{-2} = \frac{1}{4}$ ... , etc.
- $n_{10} = -b_m 2^m + \left(\sum_{i=0}^{m-1} b_i 2^i + \sum_{i=1}^{n} b_i 2^{-i}\right)$.
- Precision: 2^{-n}.
- Range: -2^m to $2^m - 2^{-n}$.
- What happens if $n = 0$?
  n=0----> Enteros son caso particular de punto fijo

En "punto fijo" la cantidad de bits de la parte decimal no cambia. En flotante si



This naming convention does not take the MSB of the number (sign bit) into account.

For instance:

- Q0.15 (Q15) 15bits+el de signo=16
  - 16 bits;
  - Range: -1 to 0.99996948;
  - Precision: 1/32768 (2^{-15}).

0 bits para parte entera, 15 para la decimal

- Q3.12
  - 16 bits;
  - Range: -8 to 7.9998;
  - Precision: 1/4096 (2^{-12}).

3 bits parte entera
12 decimal

- Q0.31 (Q31)
  - 32 bits;
  - Range: -1 to 0.999999999534339;
  - Precision: 4.6566129e-10 (2^{-31}).

Defining:

Conversor unitario

Rever esto después de ver representación en punto flotante

- Unit: $z = 1 << n = 1 \cdot 2^n$.

  Example: $n = 4 \implies z = 1.0000_2$.
- One half (1/2): $z = 1 << (n-1) = 1 \cdot 2^{(n-1)}$.

  Example: $n = 4 \implies z = 0.1000_2$.

Conversion from floating-point number x ("real") to fixed-point number X using casting:

$$X := (int)(x \cdot (1 << n)) \tag{1}$$

$$X := (int)(x \cdot 2^n) \tag{2}$$

Conversion from fixed-point number X to floating-point number x ("real") using casting:

$$x := (float)(X)/(1 << n) \tag{3}$$

$$x := (float)(X) \cdot 2^{-n} \tag{4}$$

Example 1: Represent x = 13.4 with Q4.3 format using round() function:

$$X = round(13.4 \cdot 2^3) = 107\ (01101011_2)$$

El mismo entero, según la base, vale distinto

Example 2: Represent x = 0.052246 with Q4.11 format using round() function:

$$X = round(0.052246 \cdot 2^{11}) = 107\ (0000000001101011_2)$$

- There is no difference at the CPU level (ALU) between both fixed-point and integer numbers.
- The difference is based on the concept of *scale factor*, which is completely in the head of the programmer.
- Fixed-point numbers in Qm.n notation can be seen as a signed integer simply multiplied by 2^{-n}, the precision.
- In fact, the scale factor can be an arbitrary scale that may not be a power of two.
- **Example:** We want 16-bit numbers between 8000H and 7FFFH to represent decimal values between –5 and +5.

1. Integer: -32768 to 32767 (8000H - 7FFFH).
2. $(-32768 \cdot 2^{-15})$ to $(32767 \cdot 2^{-15}) => -1$ to 0.9999948242.
3. $(-1 \cdot 5)$ to $(0.9999948242 \cdot 5) => -5$ to 4.9998741211.

   The scale factor and precision are the same, $(5 \cdot 2^{-15})$.

Dynamic range is defined as,

$$DR_{db} = 20\ log_{10}\left(\frac{\text{largest possible word value}}{\text{smallest possible word value}}\right) \quad [\text{dB}]$$

For N-bit signed integers,

$$DR_{dB} = 20\ log_{10}\left[\frac{2^{(N-1)} - 1}{1}\right] \quad [\text{dB}]$$

$$DR_{dB} \approx 20\ [(N-1)log_{10}(2)]$$

$$DR_{dB} \approx 20\ log_{10}(2) \cdot (N-1)$$

$$DR_{dB} \approx 6.02 \cdot (N-1) \quad [\text{dB}]$$

Nos da una idea de cuantos números podés representar con una cantidad de bits

Después se va a comparar este rano dinámico con el de punto flotante

Fixed point

Precision and Dynamic range examples

Format (N.M)		Largest positive value (0x7FFF)	Least negative value (0x8000)	Precision (0x0001)		DR(dB)
0	15	0,999969482421875	-1	3,05176E-05	2^-15	90,30873362
1	14	1,99993896484375	-2	6,10352E-05	2^-14	90,30873362
2	13	3,9998779296875	-4	0,00012207	2^-13	90,30873362
3	12	7,999755859375	-8	0,000244141	2^-12	90,30873362
4	11	15,99951171875	-16	0,000488281	2^-11	90,30873362
5	10	31,99902344	-32	0,000976563	2^-10	90,30873362
6	9	63,99804688	-64	0,001953125	2^-9	90,30873362
7	8	127,9960938	-128	0,00390625	2^-8	90,30873362
8	7	255,9921875	-256	0,0078125	2^-7	90,30873362
9	6	511,984375	-512	0,015625	2^-6	90,30873362
10	5	1023,96875	-1024	0,03125	2^-5	90,30873362
11	4	2047,9375	-2048	0,0625	2^-4	90,30873362
12	3	4095,875	-4096	0,125	2^-3	90,30873362
13	2	8191,75	-8192	0,25	2^-2	90,30873362
14	1	16383,5	-16384	0,5	2^-1	90,30873362
15	0	32767	-32768	1	2^-0	90,30873362

Bibliography

1. Richard G. Lyons. *Understanding Digital Signal Processing, 3rd Ed*. Prentice Hill. 2010. Chapter 12.
2. Bruno Paillard. An Introduction To Digital Signal Processors, Chapter 5.