

Finite representation of real numbers

Fixed-point numbers

Dr. Ing. Rodrigo Gonzalez

`rodrigo.gonzalez@ingenieria.uncu.edu.ar`

Control y Sistemas

Facultad de Ingeniería,
Universidad Nacional de Cuyo

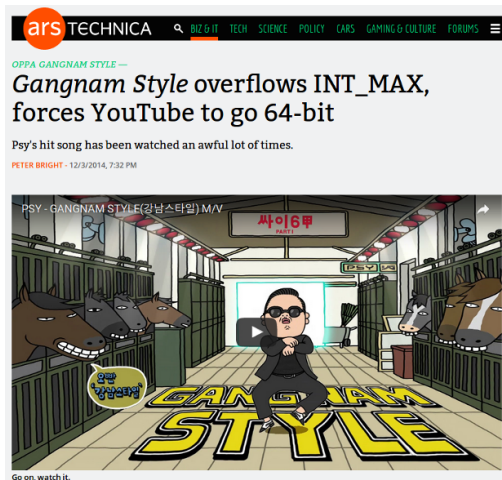


UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

Summary

- 1 Finite representation of real numbers in computers
- 2 Integers
- 3 Fixed-point
 - Scale factor
 - Dynamic range
 - How to determine the correct range
 - Addition
 - Overflow
 - How to avoid overflow
 - Accumulator
 - Multiplication
 - Underflow
 - How to avoid underflow
 - Shifts

Gangnam Style problem



<https://arstechnica.com>

Patriot Missile System problem

- On February 25th, 1991, a Patriot Missile system at Dhahran, Saudi Arabia had failed to intercept a SCUD missile. The SCUD hit an Army Barracks, killing 28 Americans soldiers.
- 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 $1/10$ th of a second in a 24-bit register.
- It results in
 $0.000111101110011001100110011001101\dots$ with an infinite number of bits.
- The error of representing $1/10$ th s 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 more than half a kilometer.
- 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.



<https://blog.penjee.com/famous-number-computing-errors/>

Integers

Unsigned integers

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

2's complement signed integers

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

in C:

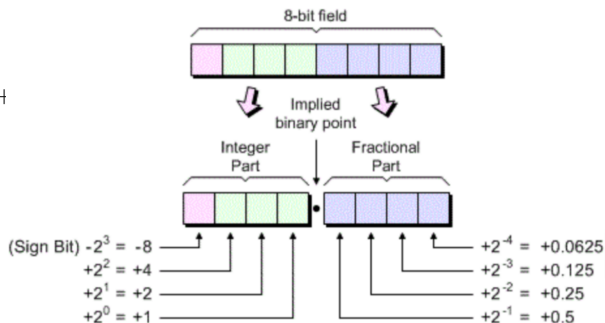
- 8 bits (`char`, `int8_t`): [-128, 127]
- 16 bits (`short`, `int16_t`): [-32768, 32767]
- 32 bits (`int`, `long`, `int32_t`): [-2147483648, 2147483647]

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

Fixed-point representation

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).
- The weights of bits to the right of the binary point are negative powers of 2: $2^{-1} = \frac{1}{2}$, $2^{-2} = \frac{1}{4} \dots$, etc.
- Precision: 2^{-n} .
- Range: -2^m to $2^m - 2^{-n}$.
- $n_{10} = (-1^{2^m}) \left(\sum_{i=0}^{m-1} b_i 2^i + \right.$



Qm.n notation

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

- Q0.15 (Q15)
 - 16 bits;
 - Range: -1 to 0.99996948;
 - Precision: $1/32768$ (2^{-15}).
- Q3.12
 - 16 bits;
 - Range: -8 to 7.9998;
 - Precision: $1/4096$ (2^{-12}).
- Q0.31 (Q31)
 - 32 bits;
 - Range: -1 to 0.999999999534339;
 - Precision: $4.6566129e-10$ (2^{-31}).

Conversion to and from fixed point

Defining:

- Unit: $z = 1 \ll n = 1 \cdot 2^n$
- One half: $z = 1 \ll (n-1) = 1 \cdot 2^{(n-1)}$

Conversion from floating-point ("real") to fixed-point number:

$$X := (\text{int})(x) \cdot (1 \ll n) \quad (1)$$

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

Conversion from fixed-point to floating-point ("real") number:

$$x := (\text{float})(X) / (1 \ll n) \quad (3)$$

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

Example: Represent $x = 13.4$ using Q4.3 format

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

Example: Represent $x = 0.052246$ using Q4.11 format

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

Scale of representation

- There is no difference at the CPU level (ALU) between a fractional and an integer representation.
- The difference is based on the concept of *scale*, which is almost completely in the head of the designer.
- Values represented 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 is not a power of two.
- Example: 16-bit 2's complement numbers between 8000H and 7FFFH can represent decimal values between -5 and $+5$, where the scale factor is $5/32768$ ($5 * 2^{-15}$).
 - Integer: -32768 to 32767 (8000H - 7FFFH).
 - Fixed point Q15: $(-32768 * 2^{-15})$ to $(32767 * 2^{-15}) \Rightarrow -1$ to 0.99996948242 .
 - $(-1 * 5)$ to $(0.99996948242 * 5) \Rightarrow -5$ to 4.99984741211 .

Scale factor, examples

Format	Scaling factor ()	Range in Hex (fractional value)
(1.15)	$2^{15} = 32768$	0x7FFF (0.99) → 0x8000 (−1)
(2.14)	$2^{14} = 16384$	0x7FFF (1.99) → 0x8000 (−2)
(3.13)	$2^{13} = 8192$	0x7FFF (3.99) → 0x8000 (−4)
(4.12)	$2^{12} = 4096$	0x7FFF (7.99) → 0x8000 (−8)
(5.11)	$2^{11} = 2048$	0x7FFF (15.99) → 0x8000 (−16)
(6.10)	$2^{10} = 1024$	0x7FFF (31.99) → 0x8000 (−32)
(7.9)	$2^9 = 512$	0x7FFF (63.99) → 0x8000 (−64)
(8.8)	$2^8 = 256$	0x7FFF (127.99) → 0x8000 (−128)
(9.7)	$2^7 = 128$	0x7FFF (511.99) → 0x8000 (−512)
(10.6)	$2^6 = 64$	0x7FFF (1023.99) → 0x8000 (−1024)
(11.5)	$2^5 = 32$	0x7FFF (2047.99) → 0x8000 (−2048)
(12.4)	$2^4 = 16$	0x7FFF (4095.99) → 0x8000 (−4096)
(13.3)	$2^3 = 8$	0x7FFF (4095.99) → 0x8000 (−4096)
(14.2)	$2^2 = 4$	0x7FFF (8191.99) → 0x8000 (−8192)
(15.1)	$2^1 = 2$	0x7FFF (16383.99) → 0x8000 (−16384)
(16.0)	$2^0 = 1(\text{Integer})$	0x7FFF (32767) → 0x8000h (−32768)

Dynamic range

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,

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

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

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

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

Precision and Dynamic range examples

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

How to determine the correct range

What is the correct value for m ?

How much bits are needed to represent $-15 \leq x \leq 10$?

MATLAB

```
❶ » INT_MIN = abs(-15); INT_MAX = 10;  
❷ » MAX = max( [ INT_MIN, INT_MAX ] ); % MAX = 15  
❸ » BITS = (log2 ( MAX ) + 2 );  
❹ » N = floor ( BITS ); % floor() redondeo a -Inf  
❺ » N = 5.00
```

Addition in 2's complement

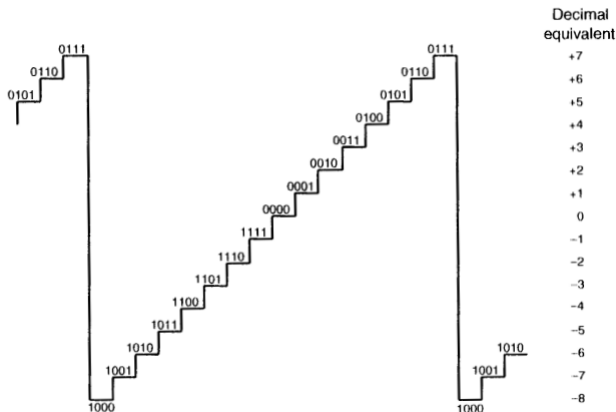
- Adding two **N-bits** numbers can produce a **N+1 bits result**.
- The result will have the same numbers of fractional bits.
- Only the integer part can grow.
- The last two bits of the carry row show if overflow occurs.

$$\begin{array}{r}
 \boxed{11}111\ 111\ (\text{carry}) \\
 0000\ 1111\ (15) \\
 +\ 1111\ 1011\ (-5) \\
 \hline
 0000\ 1010\ (10)
 \end{array}$$

$$\begin{array}{r}
 \boxed{01}11\ (\text{carry}) \\
 0111\ (7) \\
 +\ 0011\ (3) \\
 \hline
 1010\ (-6)\ \underline{\text{invalid!}}
 \end{array}$$

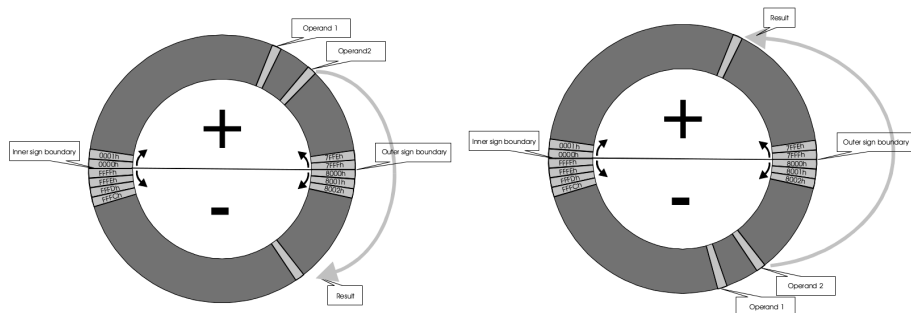
Overflow

- An **overflow** occurs in a when a result is greater than $2^{N-1} - 1$ or lesser than -2^{N-1} .
- An overflow produces a **roll-over** (wrap).



Overflow II

- A roll-over usually has catastrophic consequences on a process.
- Only happen when two very large positive operands, or two very large negative operands are added.
- It can never happen during the addition of a positive operand and a negative operand, whatever their magnitude.

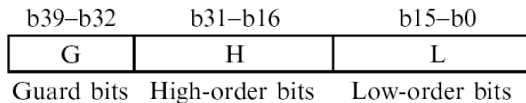


Longer word-length accumulator

- Saving the result in a $N+1$ word avoids overflows.
- The general rule is the sum of s individual m -bit can require as many as $m + \log_2(s)$.
- Example: 256 8-bits words requires an accumulator whose word length is $8 + \log_2(256) = 16$.
- DSP processors usually have 40-bit accumulators.
- How many sums are supported by a 40-bits accumulator for 16-bits numbers?

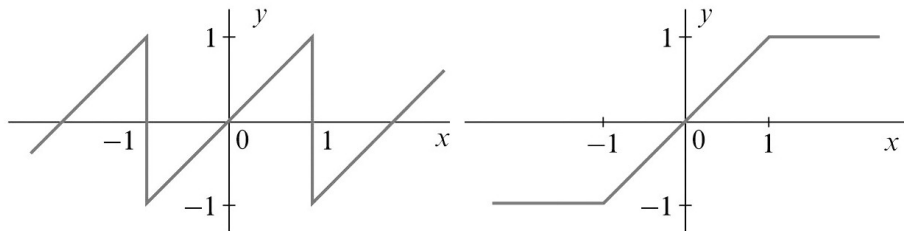
Accumulator

- DSP processors have an accumulator with extra bits to avoid overflow during internal calculations (in C5505, 40-bits accumulator).
- Guard bits: extra bits to avoid addition overflows.
- After MAC operations, only final result is adjusted to memory data size.



Saturation

- To avoid a rollover, overflow is detected and the result is saturated to the most positive or most negative value that can be represented.
- This procedure is called **saturation arithmetic**.
- DSP processors allows the results to be saturated automatically in hardware (In TI DSP C5505, SATD Bit at ST1_55 register).

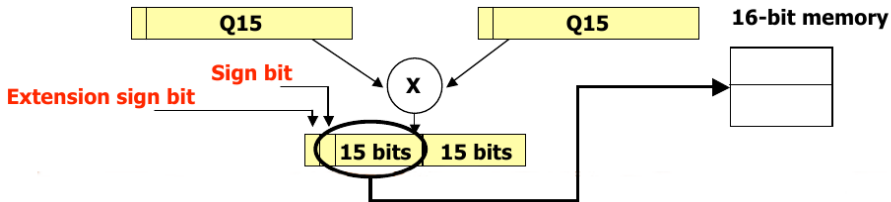


Be aware of non-linearity!

Multiplication

Multiplication in 2's complement

- The product of 2 **N-bit** numbers requires **$2 \cdot N$ bits** to contain all possible values.
- The 2 Most Significant Bits (MSB) are always equal (extension sign bit).
- Therefore, $2N-1$ bits are enough to store the result.
- A Q15 multiplication produces Q1.30 result.
- To transform the result into Q31 notation, it must be left-shifted by one bit.
- DSP processors have a special mode that allows its ALU to automatically perform the left shift when $Q15 \times Q15$.

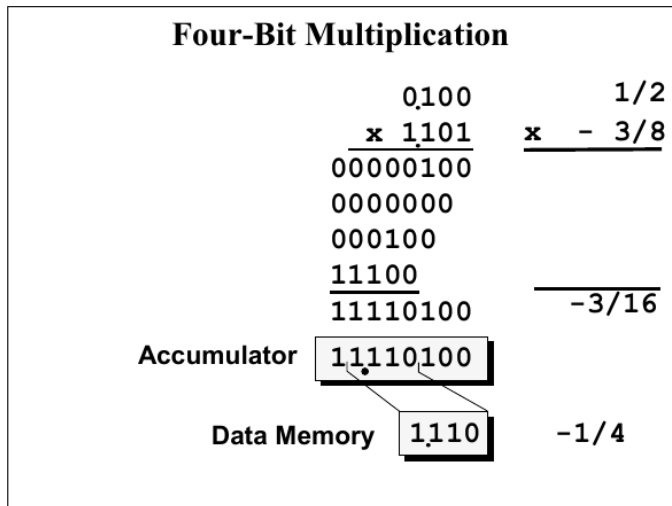


Four-bit signed integer multiplication

Four-Bit Integer Multiplication

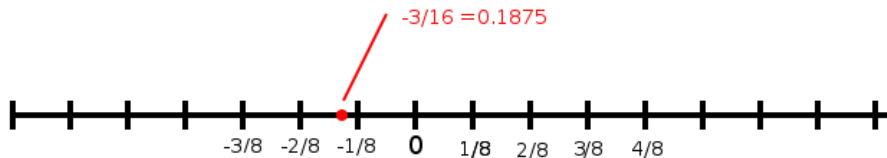
	0100	4
	<u>x 1101</u>	<u>x -3</u>
	00000100	
	0000000	
	000100	
	<u>11100</u>	
	11110100	<u>-12</u>
Accumulator	<u>11110100</u>	-12
Data Memory	<u>11110100</u>	-12

Four-bit Q0.3 multiplication



Underflow

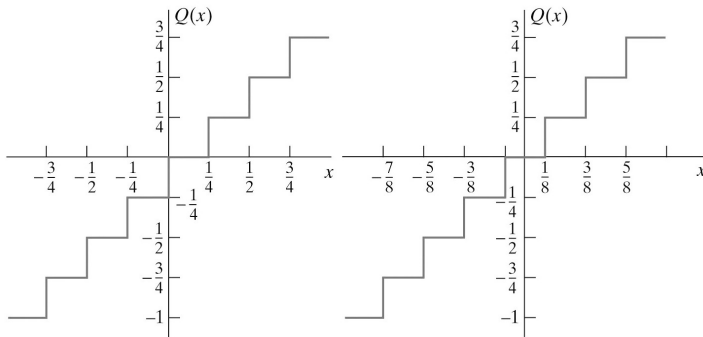
- After multiplication, $2N-1$ bits must be stored in a memory of N -bits word.
- An **underflow** occurs if a result is less than 2^{-n} .
- Q0.3 precision is $2^{-3} = \frac{1}{8}$.



- What number should the multiplication result be? $-\frac{1}{8}$ or $-\frac{2}{8}$?
- In other words, what bits should be discarded from a multiplication result?

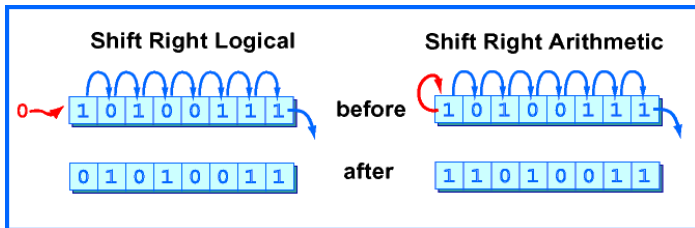
Rounding schemes, truncation and round-off

- Truncation: $e = Q[x] - x$, $-2^{-n} \leq e < 0$, $\mu = -\frac{2^{-n}}{2}$, $\sigma^2 = \frac{2^{-n}}{12}$.
- Round-off: $e = Q[x + 2^{-(n+1)}] - x$, $-2^{-n}/2 < e \leq 2^{-n}/2$, $\mu = 0$, $\sigma^2 = \frac{2^{-n}}{12}$.
- DSP processors manage truncation and round-off automatically.



Logical and Arithmetic shifts

- Multiplication: all bits are shifted left by one position.
- Division: all bits are shifted right by one position, however the sign bit must be preserved (**arithmetic shift**).
- Arithmetic shift \neq logical shift.



Logical and Arithmetic shifts II

In DSP processors:

- ALU can perform logical shifts of 32-bit operands in one cycle, from 16 bits to the right, to 15 bits to the left.
- Sign extension is performed during shifts to the right, if the Sign Extension Mode control bit (in C5505, SXM) is set.
- Result is saturated during shifts to the left if an overflow is detected, and Overflow bit (in C5505, OVM) is set.

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.