COMPUTER ARITHMETIC

AN INTRODUCTION

JEEVAN M. N.

YUVARAJA'S COLLEGE,
MYSORE

AY 2018-19

How do computers work?

BITS AND BYTES

- A computer understands numbers in terms of "HIGH" and "LOW" voltages, say +5V and 0V which correspond to the binary digits "1" and "0" respectively.
- A single binary digit is called a "BIT".
- A sequence of 8 arbitrary binary digits is called a "BYTE".

A COMPUTER WORD

- A computer word is a natural unit of data used in a computer architecture.
- The number of bits in a computer word is known as it's "word length".
- Some common computer architectures used today have computer words of 16 bit length, 32 bit length and 64 bit length.

HOW COMPUTERS UNDERSTAND NUMBERS

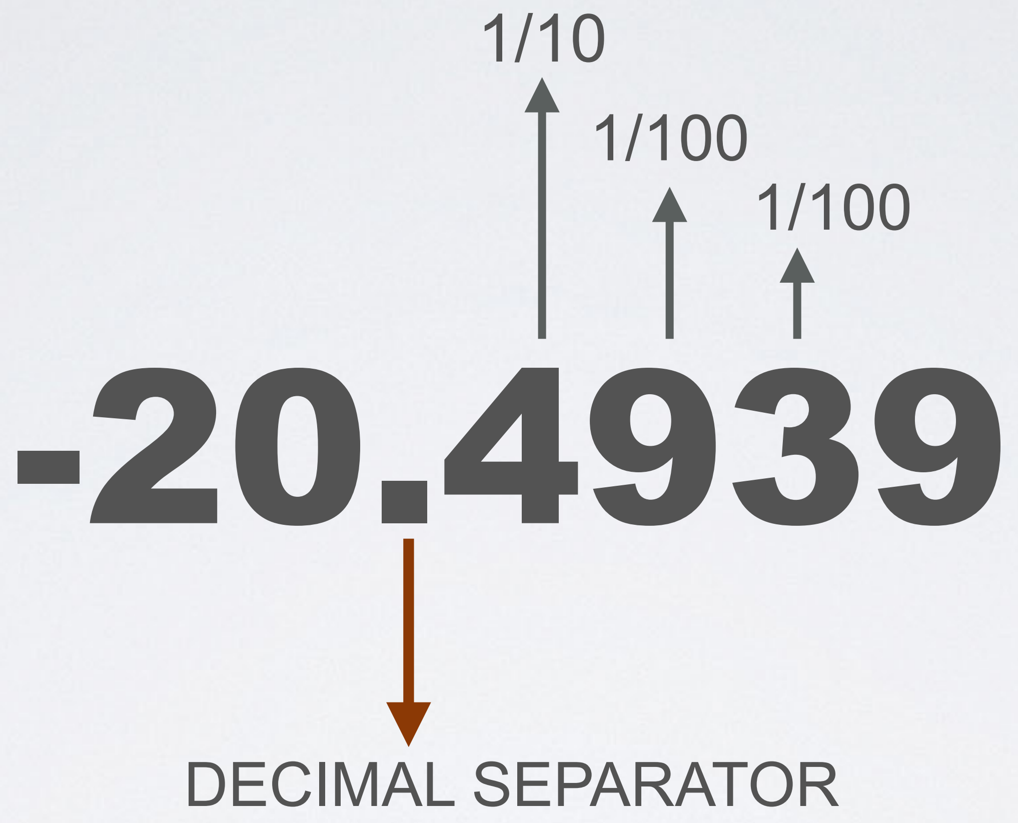
- Computers are designed to do calculations using electrical signals and the binary number system is the convenient one to use.
- Humans are familiar with the Decimal number system.
- Some other number systems are the Hexadecimal and Octal systems.

1. NUMBER SYSTEMS

DECIMAL / BASE-10 NUMBER SYSTEM

- The *Decimal System* Assigns a finite sequence of symbols taken from the set D = { 0,1, 2, 3, 4, 5, 6, 7, 8, 9 } known as "digits" to non zero whole numbers such that the first digit is always non-zero.
- Negative numbers are represented by a minus " - " sign appearing before the sequence.
- The "Decimal point / dot" is employed to write fractions of whole numbers, for example: 3/2 = 1.5
- The position of a digit in the sequence denotes it's "Place value".

A DECIMAL NUMBER



$$-(2\times10^{1} + 0\times10^{0} + 4\times10^{-1} + 9\times10^{-2} + 3\times10^{-3} + 9\times10^{-4}) = -20.4939$$

An example of "Positional Notation"

THE BINARY NUMBER SYSTEM

- The *Binary number system* employs a sequence of symbols taken from the set B = { 0, 1 } to denote a whole number.
- A specified digit in the sequence, say the first or the last, may be used to denote negative / positive numbers. (Signed Integer)
- example of a decimal to binary conversion:
  47 = 0101111

THE FIRST FEW BINARY NUMBERS

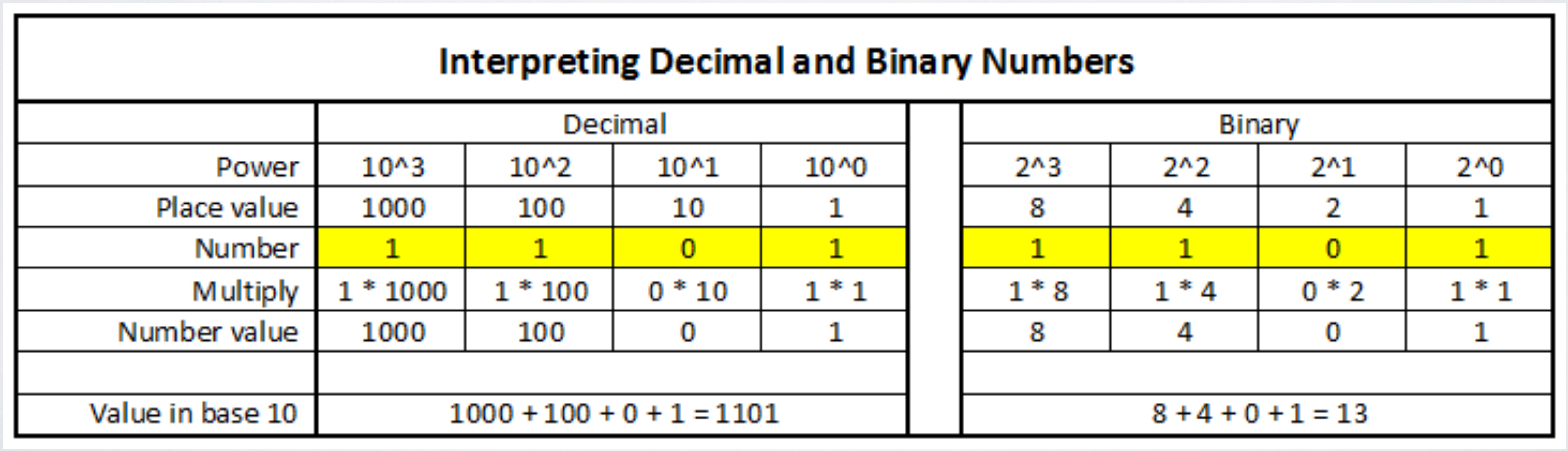
0	=	0	110	=	6
1	=	1	111	=	7
10	=	2	1000	=	8
11	=	3	1001	=	9
100	=	4	1010	=	10
101	=	5	1011	=	11

Example 1

$$19 = 1\text{x}2^4 + 0\text{x}2^3 + 0\text{x}2^2 + 1\text{x}2^1 + 1\text{x}2^0$$

$$19 = 1 \quad 0 \quad 0 \quad 1 \quad 1$$

Example 2



Interpreting Decimal and Binary Numbers

	Decimal					Binary			
Power	10^3	10^2	10^1	10^0		2^3	2^2	2^1	2^0
Place value	1000	100	10	1		8	4	2	1
Number	1	1	0	1		1	1	0	1
Multiply	1 * 1000	1 * 100	0 * 10	1 * 1		1 * 8	1 * 4	0 * 2	1 * 1
Number value	1000	100	0	1		8	4	0	1
Value in base 10	1000 + 100 + 0 + 1 = 1101					8 + 4 + 0 + 1 = 13			

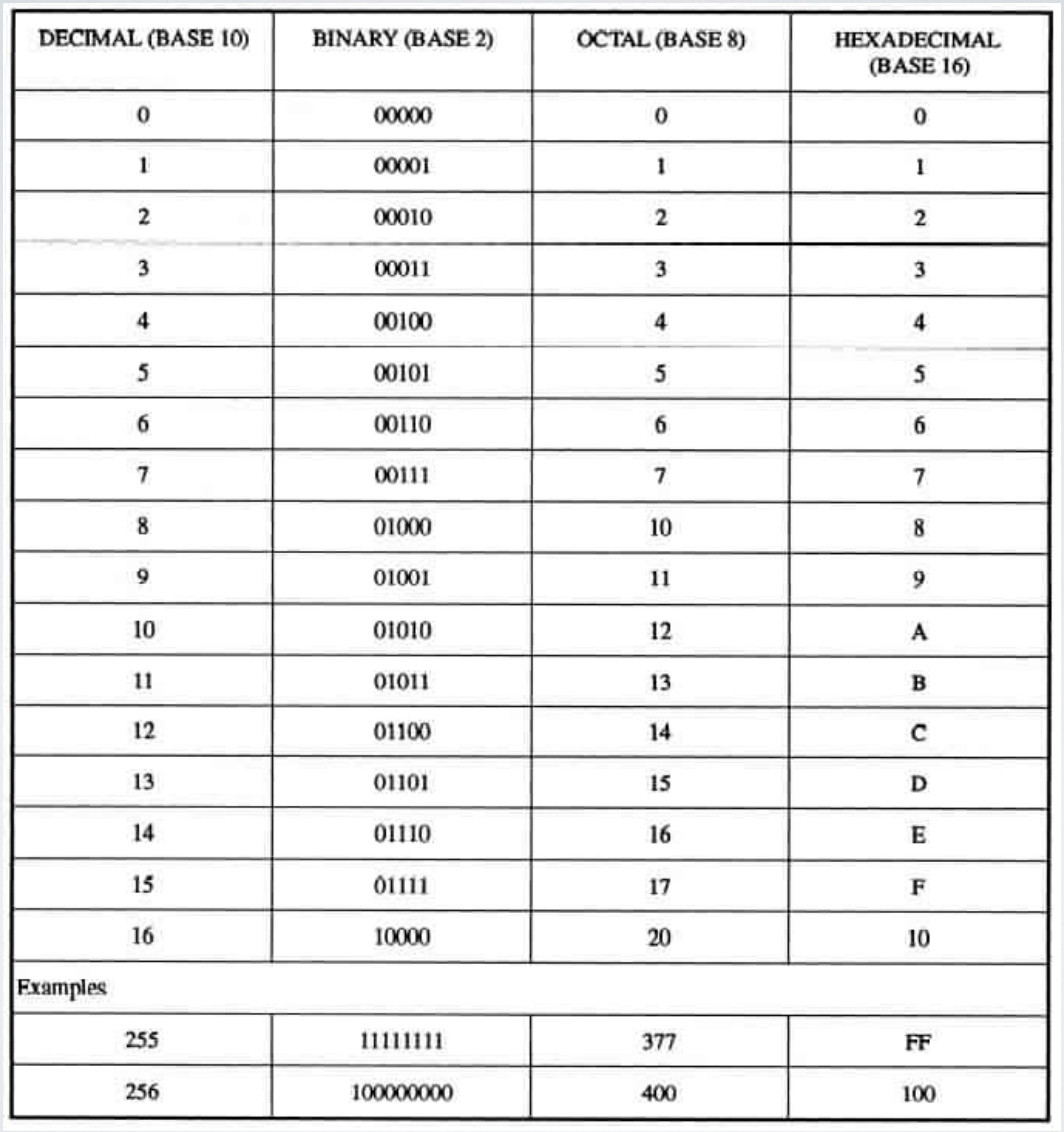
Image: (CC-BY-SA) TychaBrahe, wikimedia commons

CONVERTING A DECIMAL NUMBER TO A BINARY NUMBER
A SIMPLE ALGORITHM

1. Find the largest power of 2 that lies within the given number.
2. Subtract that value from the given number.
3. Find the largest power of 2 within the remainder found in step 2.
4. Repeat until there is no remainder.
5. Enter a 1 for each binary place value that was found, and a 0 for the rest.

ARITHMETIC OPERATIONS ON BINARY NUMBERS

- Adding Binary numbers.
- Subtracting Binary numbers.
- Multiplying Binary numbers.
- Division of Binary numbers.



DECIMAL (BASE 10)	BINARY (BASE 2)	OCTAL (BASE 8)	HEXADECIMAL (BASE 16)
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
Examples			
255	11111111	377	FF
256	100000000	400	100

2. THE PROBLEM

THE PROBLEM

- Let's add two binary numbers

$$
\begin{array}{lc:cccc l}
 & 1 & 1 & 1 & 0 & & \text{CARRY} \\
\text{Value 1:} & & 1 & 0 & 1 & 1 & \\
\text{Value 2:} & + & 0 & 1 & 1 & 0 & \\
\hline
\text{Result:} & 1 & 0 & 0 & 0 & 1 &
\end{array}
$$

- What if my computer can understand only 4-bit signals ?
- This error is known as "Overflow error" and occurs with other arithmetic operations too.

THE PROBLEM WITH REAL NUMBERS

- The square root of 3 is an irrational number.
- In the decimal system we have a non terminating sequence of digits: $\sqrt{3} = 1.7320508..........$
- In the Binary system too we need to assign a non-terminating sequence of 1s and 0s to represent this number.

HOW MANY NUMBERS CAN A BYTE STORE?

- A byte is a sequence of only 8 binary numbers and it cannot accommodate all the real numbers, let alone natural numbers.

2 bits can address $2^2 = 4$ numbers
4 bits can address $2^4 = 16$ numbers
8 bits can address $2^8 = 256$ numbers
16 bits can address $2^{16} = 65536$ numbers

- How can the amount of numbers stored in a byte be increased?

What we need is a method to accommodate as many numbers as possible in a given bit length and account for the errors that might arise due to the fact that computers can only deal with a finite number of bits.

3. NUMERICAL ERRORS

SIGNIFICANT FIGURES

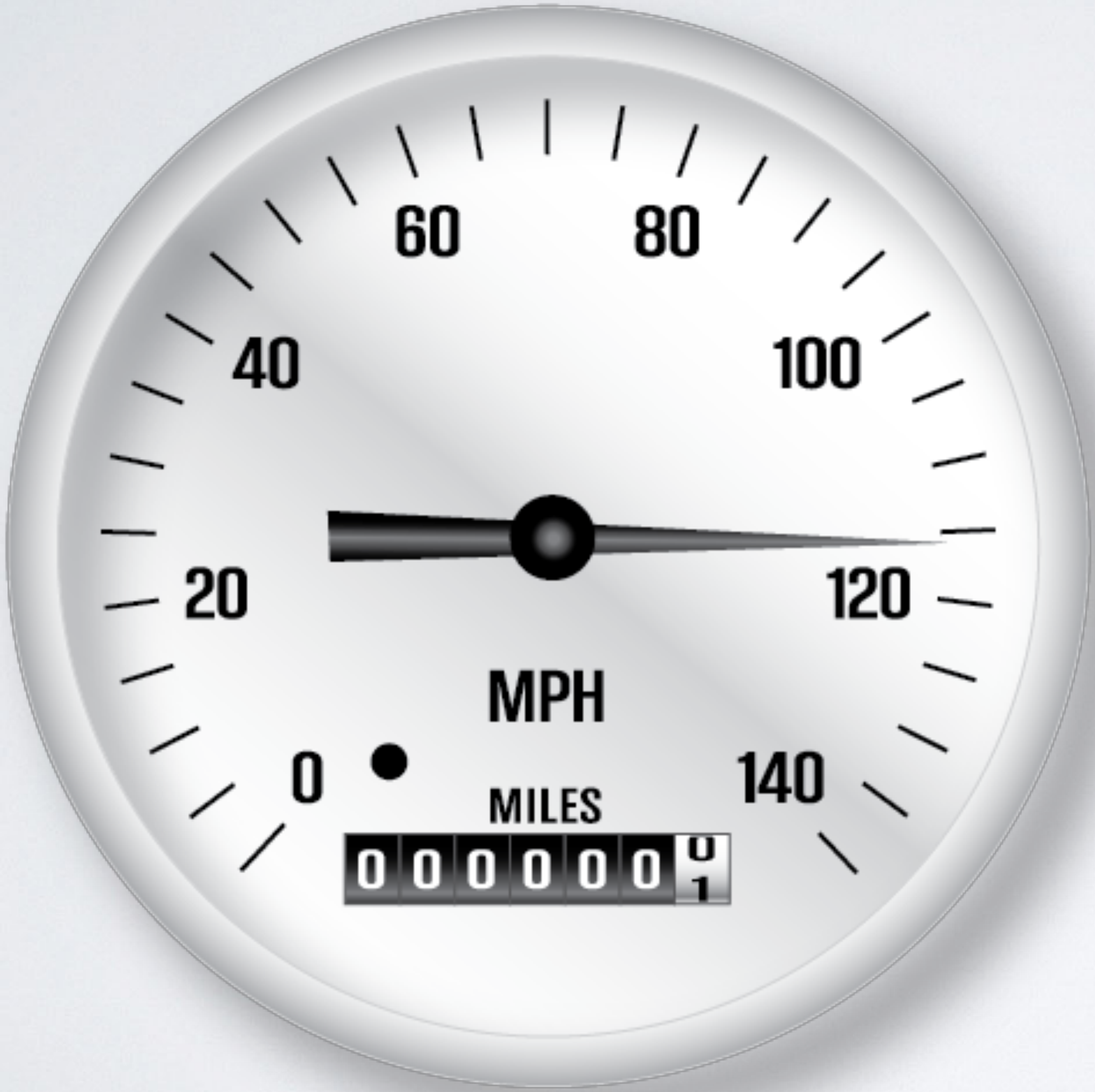


Image: pngimg.com | Creative Commons

What is the reading shown on the speedometer?

SIGNIFICANT FIGURES / DIGITS

- The significant figures/digits of a number are those that can be used with confidence.
- In the previous example we can say with assurance that the vehicle is moving with a velocity greater than 115 miles/hour. But we cannot know the exact velocity due to the limitations of the measuring apparatus.
- The number of significant digits of a number measured in an apparatus are the "certain digits" plus an "estimated digit". Conventionally, the estimated digit is set to one-half of the smallest scale division on the measurement device.
- Thus the significant digits in the previous example are 115 + (5/2) = 117.5 i.e 4 significant digits.

NUMERICAL ERRORS

Numerical errors arise from the use of approximations to represent exact mathematical operations and quantities.

- TRUNCATION ERRORS
- truncation errors occur when approximations are used to represent exact mathematical procedures.
- example: approximating a derivative by a ratio of finite quantities.
- $\dfrac{dx}{dt} \cong \dfrac{\Delta x}{\Delta t} = \dfrac{x_{i+1} - x_i}{t_{i+1} - t_i}$

- ROUND-OFF ERRORS
- Round-off errors occur when numbers having limited significant digits are used to represent exact numbers
- example: 12.4594 is used to represent the exact number 12.4594892241
- …..

ROUNDING OFF A NUMBER

- A number can be rounded off in two ways, either by “chopping” or by “rounding”:

$$\sqrt{5} = 2.2360679775$$

- Chopping to four decimal places: 2.2360
- Rounding to four decimal places: 2.2361
- Chopping and rounding introduce errors as the number cannot be represented exactly.

ERRORS

$$True\ value = approximation + error$$

$$E_t = Truevalue - approximation$$

$$(True)\ Fractional\ Relative\ Error = \frac{True\ Error}{True\ Value}$$

$$(True)\ Percent\ Relative\ Error$$

$$\varepsilon_t = \frac{True\ Error}{True\ Value} \times 100\,\%$$

APPROXIMATE ERRORS

Approximate percentage error in an iterative approach

$$\varepsilon_r = \frac{Current\ approximation\ -\ Previous\ approximation}{Current\ approximation} \times 100\,\%$$

4. THE SOLUTION ?

TRYING DIFFERENT STRATEGIES

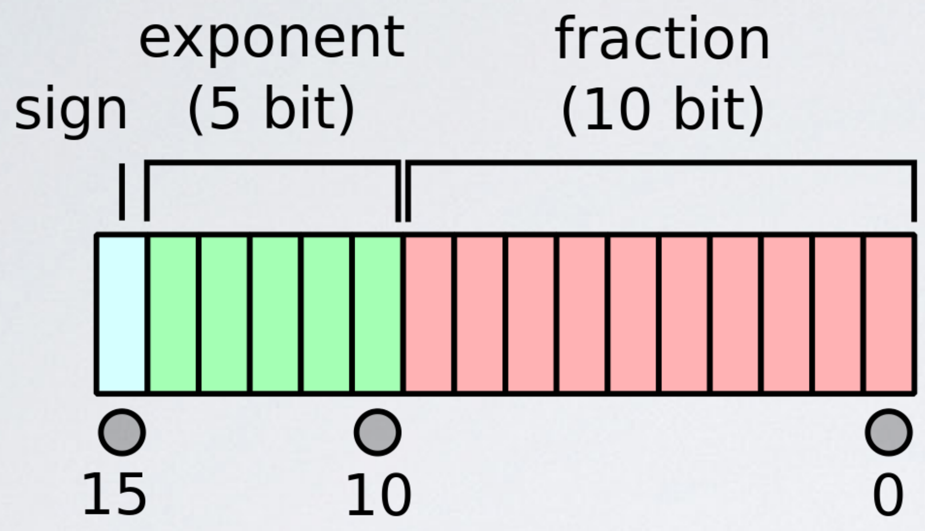
Amount of numbers stored in a computer word when:

1. Each sequence addresses a whole number.
2. Each sequence address an integer.
3. The first bit denotes the sign and the remaining bits address a whole number.
4. The first eight bits address the number appearing before the decimal point and the remaining bits address the number appearing after the decimal point.
5. The first bit addresses the sign of the number, the next 3 bits address the number in the exponent of 10 and the remaining bits address whole numbers.
6. The first bit addresses the sign of the number, the next bit addresses the sign of the exponent, the next two bits address the magnitude of the exponent and the remaining bits address a whole number.

FLOATING POINT NUMBERS

- Divide a number into 2 parts, i.e mantissa and exponent.
- $$m \cdot b^e$$
- A part of the memory address is assigned to the mantissa (m) and the remaining part is reserved for the sign of the number and it’s exponent (e). Here b denotes the base of the number system.
- The amount of real numbers that can be stored in a memory address/ bit sequence will be much larger than the case where each finite sequence is assigned to a single number.

FLOAT/ FLOATING POINT NUMBERS



IEEE 16 BIT FLOATING POINT NUMBER FORMAT

FIXED POINT NUMBERS	FLOATING POINT NUMBERS
• Assigns half of the memory address to the sequence appearing before the decimal point and the remaining half for the number after the decimal point.	• Assigns a part of the memory address to the sign, another part to the signed exponent and the remaining part to the significant digits.

How many numbers can we accommodate in a computer word and how accurately can we represent an arbitrary real number?

AN ILLUSTRATIVE EXAMPLE

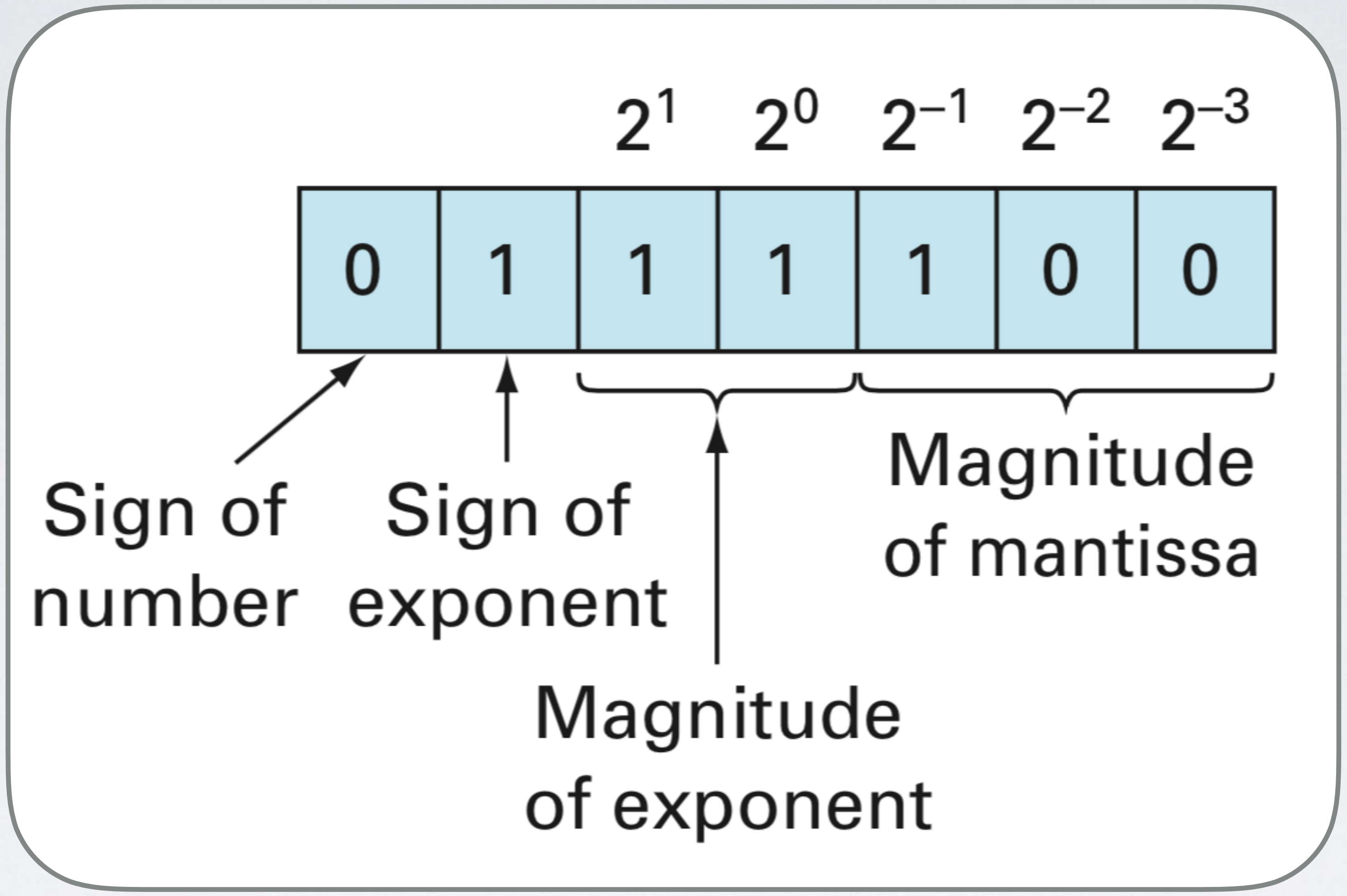
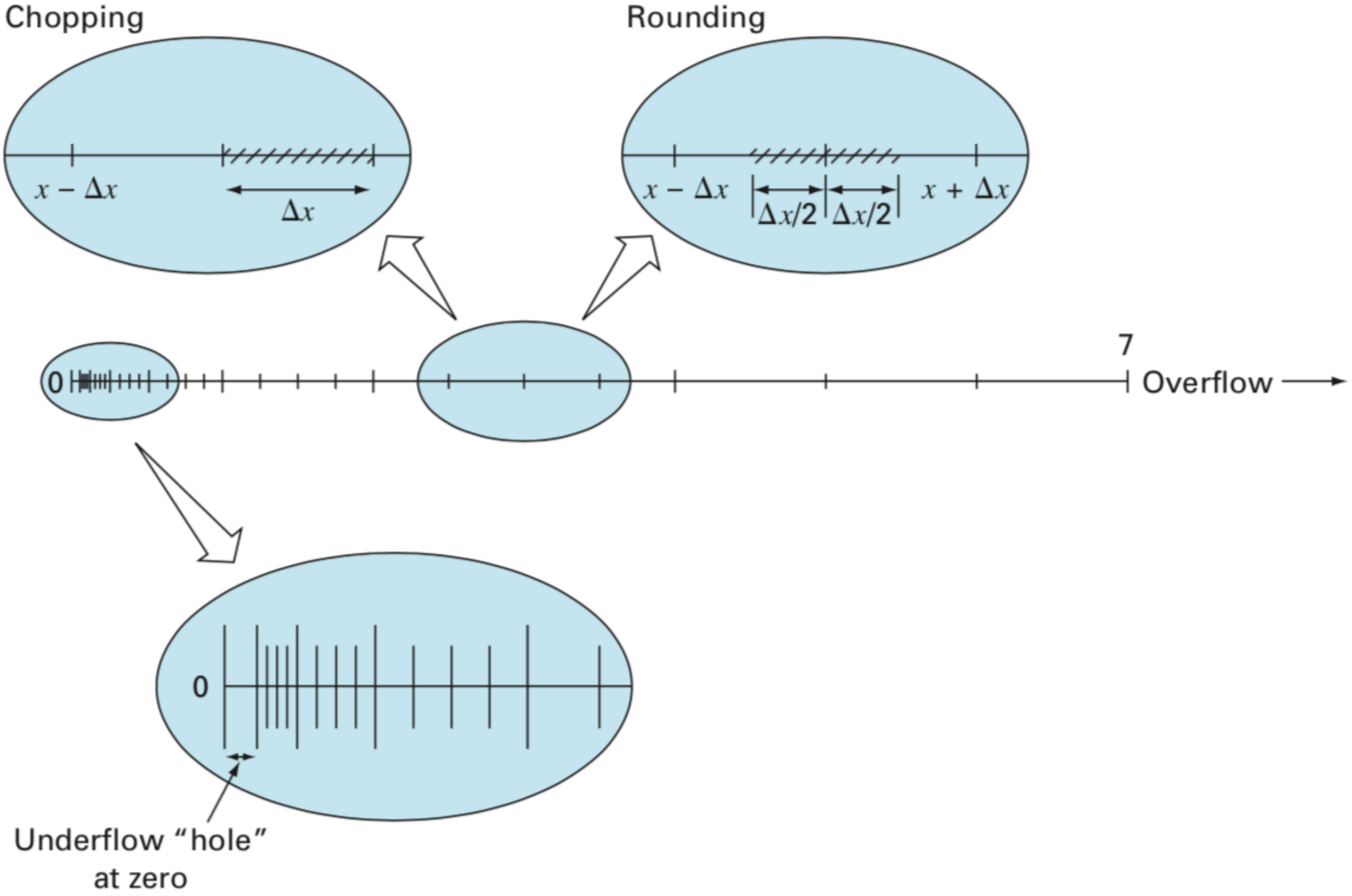


Image: Numerical methods for engineers | S.C.Chapra and R.P.Canale

Working with a 7-bit computer word



Quantising Errors

Image: Numerical methods for engineers | S.C.Chapra and R.P.Canale

OVERFLOW AND UNDERFLOW

OVERFLOW error occurs when the result of a calculation is a number which is the larger than the largest number that can be stored in a given computer word.

UNDERFLOW error occurs when the result of a calculation is a number which is smaller than the smallest number that can be stored in a given computer word.

THE PROBLEMS THAT PERSIST

1. The range of quantities that are represented by a computer word are limited.

2. There are only a finite number of quantities that can be represented within a range.

3. The interval between numbers increases as the numbers grow in magnitude.

SOME COMMON DATATYPES

INT stores integers.

LONG stores integers which are longer than the datatype INT can store.

FLOAT stores all real numbers.

DOUBLE stores double-precision floating point number values.

STRING stores alphanumeric characters as text.

THANK YOU

Presentation created by Jeevan M N,
Yuvaraja's College, Mysore
AY 2018-2019

All images belong to the respective copyright holders.

Appendix 1

Normalisation

1. In a normalised decimal or binary number:
   $1 \leq \mathit{Mantissa\ (Significand)} < \mathit{base}$
2. The exponent must be an integer.
3. Examples:

   These are normalised:
   - $+1.23456789 \times 10^{1}$
   - $-9.987654321 \times 10^{12}$
   - $+5.0 \times 10^{0}$

   These are not normalised:
   - $+11.3 \times 10^{3}$ mantissa > base
   - -0.0002×10^{7} mantissa < 1
   - $-4.0 \times 10^{1/2}$ exponent not an integer

Appendix II
Converting numbers between Number Systems

Binary to Decimal

1.00101_2 $= 1\times 20 + 0\times 2\text{-}1 + 0\times 2\text{-}2 + 1\times 2\text{-}3 + 0\times 2\text{-}4 + 1\times 2\text{-}5$

$= 1 + 0/2 + 0/4 + 1/8 + 0/16 + 1/32 = 1 + 0.125 + 0.03125$

$= 1.5625_{10}$

Decimal to Binary

Let's start with $3.4625 \times 10^1 = 34.625$

Let's deal separately with the 34 (which equals 100010_2)

$2 \times 0.625 = 1.25$ (save the integer part)

$2 \times 0.25 = 0.5$ (no integer part to save)

$2 \times 0.50 = 1.00$ (save the integer part)

Let's write them left to right in order: $34.625_{10} = 100010.101_2$

References

1. Numerical methods for engineers- Steven. C. Chapra and Raymond. P. Canale. 7th ED.
2. Elementary numerical analysis- Kendall Atkinson. 3rd ED.