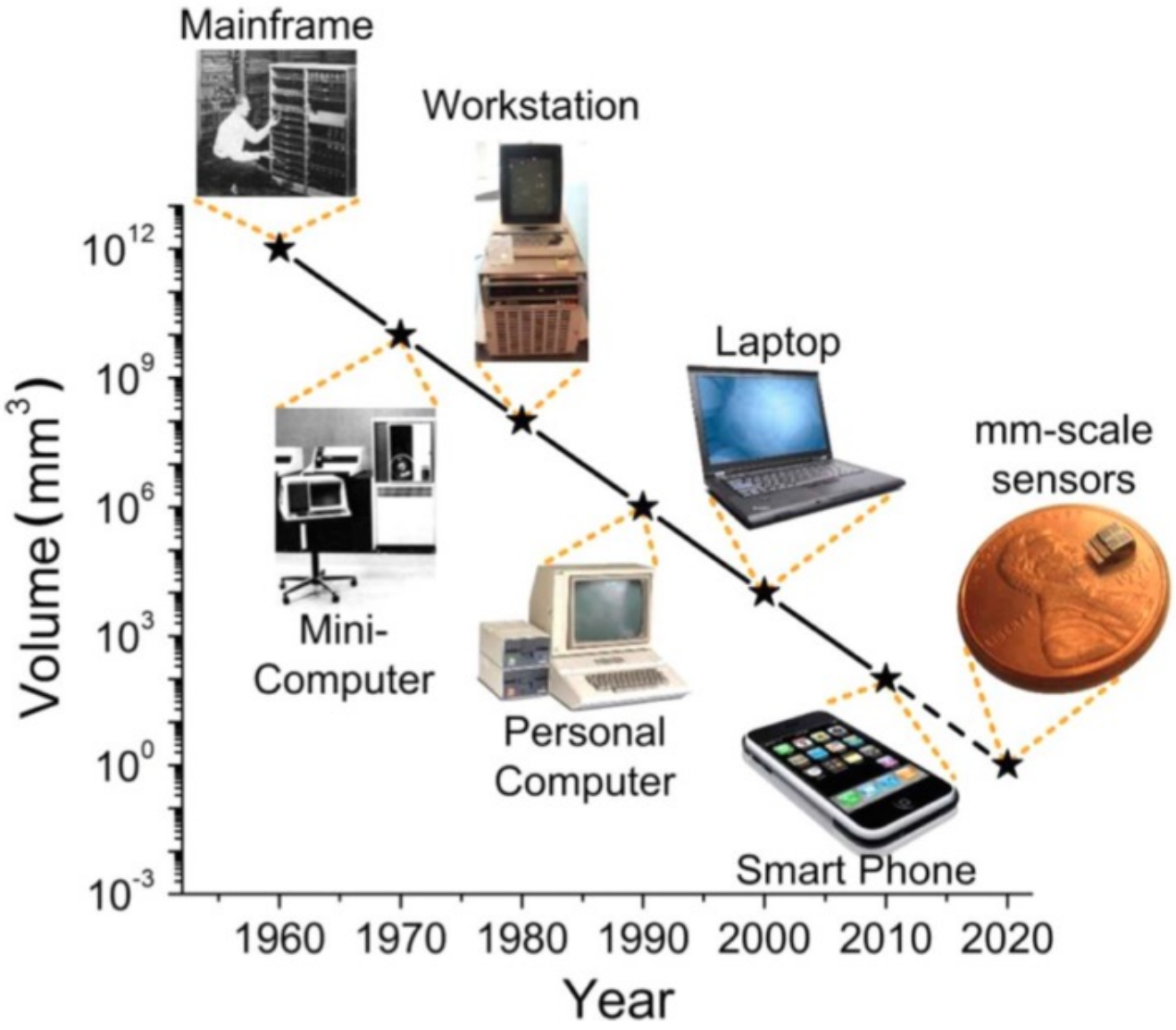
INTRODUCTION TO COMPUTATION-I

CBCS Honours Course (PHYSICS)
B.Sc 1st Semester
Paper: Mathematical Physics Lab
PHY-HC-1016

Prepared by
Dr. Ananya Phukan
Department of Physics
Mangaldai College

WHAT IS A COMPUTER?

A computer is an advanced electronic machine that takes raw data from the user as **input**, **stores** and processes it following a set of instructions that produces some **result (output)**



Hardwares

Input Device

Central Processing Unit

Control Unit

Arithmetic/Logic Unit

Memory Unit

Output Device

COMPUTER ARCHITECTURE

- Modern computers are based on *von Neumann Architecture*.
- Basic structure of a computer should consist the following components:
  - Input/Output Unit (I/O)
  - Arithmetic Logic Unit (ALU)
  - Control Unit (CU)
  - Storage / Memory Unit
  - Bus

COMPUTER ARCHITECTURE

- **Input Unit**

Input unit takes data and instructions from the user to interact with the computer system.

Example: Keyboard, Mouse, Scanner, etc.

- **Output Unit**

It provides / displays the result of the computation – the data and instructions to the user.

Example: Monitor, Printer, Speakers, etc.

COMPUTER ARCHITECTURE

- **Arithmetic Logic Unit (ALU)**

All the mathematical and logical calculations are performed by ALU. When the operations are done, the result is transferred to the storage unit.

- **Control Unit**

It controls the flow of data and instructions from ALU to storage unit and back.

- **CPU: Central Processing Unit**

The Arithmetic Logic unit and the Control Unit together is called the Central Processing Unit (CPU). It is like the brain of the computer!

COMPUTER ARCHITECTURE

- **Storage Unit**

This unit holds the data and instructions. Input data and intermediate results are stored for any future use. There are two main categories of the storage or memory units:

- **RAM: Random Access Memory**

  This is used for temporary storage of data. When the computer is switched off, the data is lost. It is used as the primary storage.

- **ROM: Read Only Memory**

  This is used for permanent storage of data. ROM is treated as secondary memory. The devices are Hard Disk, CD, DVD etc. The secondary memory is slower and cheaper than the primary memory.

COMPUTER ARCHITECTURE

- **Bus**
  - Buses are the communication channels (wiring plus connectors) that connects the processor to primary storage and input/output devices.
  - Communication on a bus is broken into discrete *transactions*. Each transaction has a sender and a receiver.
  - The performance of a bus is defined by two parameters, the transfer time, and the overall *bandwidth* (sometimes called *throughput)*. Bandwidth is expressed in units of bits per second (bps), measures the capacity of the bus.

SOFTWARE

- To make the Computers do their job means to instruct the processors. But the processors do not understand human language. So, there are various computer or programming languages evolved which are compiled or translated or interpreted to interact with the machine.
- Types of computer languages:
  - **Low- level language**
    1. ***Assembly language***
    2. ***Machine language***
  - **High- level language**

LOW-LEVEL LANGUAGE

- Low-level languages are closer to the way a machine works.
- Electronic processors in a computer are made of logic gates (or switches), which can understand only signals low and high. So, the signals of two kinds, let us say 1 and 0, make a binary code such as 110110100. The sequence of such codes when fed into a computer converts into electrical signals.
- **Machine Language**
  - It basically uses binary numbers, i.e. 0's and 1's
  - A computer can directly understand machine language. It doesnot require a compiler or inyterpreter. It works faster. Details of hardware architecture is needed.
  - It is difficult to handle a large code in 0's and 1's for common users. Difficulty in fixing errors or *debugging*.

LOW-LEVEL LANGUAGE

- **Assembly language**
  - Assembly languages was constructed with a set of letters and symbols in order to improve upon the machine language.
  - 'Assembler' is a translator program that converts an assembly language to machine language. It is called a *second-generation language*.
  - Easier to handle as compared to machine language
  - Disadvantage is that an assembly language is still machine dependent. So, the programs in assembly language written for one computer might not work for other computer.

HIGH – LEVEL LANGUAGE

- High-level languages carry out instructions given in English words following the logical steps.
- It consists of simple English words, numbers and some mathematical and other symbols [+ , - , * , / , % , ; , {} , () …….]
- Easy to understand and independent of computer hardware architecture.
- Examples: **BASIC** (Beginners All-purpose Symbolic Instruction Code), **COBOL** (Common Business Oriented Language), **FORTRAN** (Formula Translation), **C**, **C++**, **JAVA**, **PASCAL, HASKELL, RUBY, PEARL, PYTHON**

BASICS OF SCIENTIFIC COMPUTING

- Binary and decimal arithmetic
- Floating point numbers
- Algorithms
- Sequence, Selection and Repetition
- Single and double precision arithmetic
- Underflow & overflow- emphasize
- Importance of making equations in terms of dimensionless variables
- Iterative methods

NUMBER SYSTEMS

- Decimal Number System (Base-10, Digits used: 0 to 9)
- Binary Number System (Base-2, Digits used: 0 and 1)
- Octal Number System (Base-8, Digits used: 0-7)
- Hexadecimal Number System (Base-16, Digits used: 0-9, Letters used : A to F)

DECIMAL NUMBER SYSTEM

- The number system that we use in our day-to-day life is the decimal number system.
- Decimal number system has base 10 as it uses 10 digits from 0 to 9.
- In decimal number system, the successive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on.

Example: The number 1234 can be written as

$(1\times1000) + (2\times100) + (3\times10) + (4\times l)$

$(1\times10^3) + (2\times10^2) + (3\times10^1) + (4\times l0^0)$

$1000 + 200 + 30 + 1$

1234

BINARY NUMBER SYSTEM

- Uses two digits, 0 and 1.
- Also called base 2 number system.

Example: The decimal equivalent of 10101 is

Step	Binary Number	Decimal Number
Step 1	10101_2	$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	10101_2	$(16 + 0 + 4 + 0 + 1)_{10}$
Step 3	10101_2	21_{10}

Note: 10101_2 is normally written as 10101.

BINARY ADDITION

Case	A+B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

Example:

$0011010 + 001100 = 00100110$

```
  1 1          carry
  0 0 1 1 0 1 0   = 26₁₀
+ 0 0 0 1 1 0 0   = 12₁₀
-----------------
  0 1 0 0 1 1 0   = 38₁₀
```

BINARY SUBTRACTION

Case	A+B	Subtract	Borrow
1	0 - 0	0	0
2	0 -1	1	0
3	1 - 0	0	0
4	1 - 1	0	1

Example:

0011010 - 001100 = 00001110

1 1	borrow
0 0 ~~1 1~~ 0 1 0	$= 26_{10}$
- 0 0 0 1 1 0 0	$= 12_{10}$
0 0 0 1 1 1 0	$= 14_{10}$

BINARY MULTIPLICATION

Case	A . B	Multiplication
1	0 . 0	0
2	0 . 1	0
3	1 . 0	0
4	1 . 1	1

Example:

0011010 x 001100 = 100111000

$$
\begin{array}{rl}
0011010 & = 26_{10} \\
\times\, 0001100 & = 12_{10} \\
\hline
0000000 & \\
0000000 & \\
0011010 & \\
0011010 & \\
\hline
0100111000 & = 312_{10}
\end{array}
$$

BINARY DIVISION

101010 / 000110 = 000111

```
              1 1 1        = 7₁₀
        _______________
000110 ) 1 0 1 0 1 0       = 42₁₀
        -1 1 0             = 6₁₀
         -------
           1 0 0 1
          -1 1 0
          --------
             1 1 0
            -1 1 0
           --------
                 0
```

BOOLEAN ALGEBRA

- Binary numbers follow a different kind of algebra, known as **Boolean Algebra.**
- Boolean Laws:

$0 + 0 = 1, 1 + 1 = 1, 1 + 0 = 0 + 1 = 1$ (OR Gate)

$0 \,.\, 0 = 0, 1 \,.\, 1 = 1, 1 \,.\, 0 = 0 \,.\, 1 = 0$ (AND Gate)

$\bar{1} = 0, \bar{0} = 1$ (NOT Gate)

AND, OR, NOT are logic gates that the computer processors are made of.

BIT & BYTE

- A single digit (0 or 1)in a binary number is known as **bit.**
- A string of 8 digits (8 bits) is termed as **byte.**

  1 byte = Collection of 8 bits [For example: 11001001, 11110011, …….]
- A ‘byte’ is commonly defined as a unit of computer memory. One byte can store one **character, e.g., ‘X’ or ‘@’ or ‘m’.**
- 1 **KB** (Kilobyte) = 1024 bytes (eg. A small email)

  1 **MB** (Megabyte) = 1024 Kilobytes (eg. MP3 files)

  1 **GB** (Gigabyte) = 1024 Megabytes

  1 **TB** (Terabyte) = 1024 Gigabytes

STORING AN INTEGER

Bit Position

	3	2	1	0
Decimal Number 0	0	0	0	0
Decimal Number 1	0	0	0	1
Decimal Number 2	0	0	1	0
Decimal Number 3	0	0	1	1

In a **signed binary number** (positive or negative) on a 1-byte (i.e 8-bit) processor, the **left most bit** is for sign and the rest are for magnitude

+14	**0**	0	0	0	1	1	1	0
-14	**1**	0	0	0	1	1	1	0

WHAT IS A 32 BIT AND 64 BIT PROCESSORS?

- Most Desktop Computers (PC) and laptops have 32- bit (4 bytes) and 64-bit (8 bytes) processors.
- The computer processor can process a 32 bit or 64 bit long binary string at a time. So, to the computer the word length is 32 bits or 64 bits.
- The number of bits processed at a time is called a **computer word**, where the bits are the ‘letters’ for a computer word.

INTEGER OVERFLOW

- Integers are stored with either 4 bytes (32 bits) or 8 bytes (64 bits). For example, with 4 bytes, we can store $2^{32} = 4294967296$ different numbers, counting from 0 to 4294967296, if all positive.
- For accommodating both positive and negative numbers, we can store numbers between -2147483648 to 2147483647.
- The maximum positive number a 32-bit processor can handle is 2147483647. So even if we add 1 more to it, the number goes to -2147483648, the other end. This is called **Integer Overflow.**

FLOATING POINT NUMBERS / FLOAT

- Numbers with decimal places (fractional part) are called **floating point numbers** or **float.**
- For example: **0, 3.14, 6.5,** and **-125.5** are all Floating Point numbers.
- Compared to Floating Point numbers Integers are precise and there can never be any rounding errors.
- Integer numbers can be stored by just manipulating bit positions.
- The dot (.) used to separate the fractional part from the integer part is called **radix point**. In decimal number system, it is known as **decimal point**. Similarly, in binary number system, it is called **binary point**.
- To represent a fractional binary number, we must consider three parts:
    i. the bits that represent the **integer part**
    ii. the bits that represent the **fractional part**
    iii. the sign field (+/-)

STORING FLOATING POINT

The binary point on the baseline for a binary float **1101.11**

Power of 2	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}
Binary digit	1	1	0	1	.	1	1

REPRESENTATION OF FRACTIONAL NUMBERS

- In computer technology, the **real numbers** are the numbers with fractional parts. A real number can be represented as, for example, $\mathbf{13.65 = 1365 \times 10^{-2}}$, where the leading part $\mathbf{1365}$ is **significand** with the base 10 with **exponent -2**
- With this representation, we can clearly see the order of magnitude of a number by looking at the exponent. For very large numbers (eg: distance between Galaxies) or for very small numbers (eg: the dimension of an atomic nucleus), this notation is very effective.
- For floating point operations, a computer has **Floating Point Unit (FPU)**, a mathematical coprocessor, which is specially designed to do this job. The floating-point operations performed per unit time by the FPU unit are measured in terms of **FLOPS**.

STORAGE FORMATS

- **Fixed Point Representation:** A specific number of bits are fixed for integer part, and a specific number of bits are fixed for fractional part.

Eg: In a 32-bit long string, 1 bit is reserved for sign (0 for + and 1 for -), 15 bits for integer and 16 bits for fractional part. The binary number -101011.101 is represented as-

1 | 000000000101011 | 101000000000000

So, in the above format, the largest number that can be represented is,

0 | 111111111111111 | 111111111111111

In decimal,

$$+\left(\mathbf{1\times2^{14}+1\times2^{13}+\ \ldots\ldots\ldots+1\times2^{0}}\right)+\left(\mathbf{1\times2^{-1}+1\times2^{-2}+\ \ldots\ldots\ldots+1\times2^{-16}}\right)$$

The smallest positive number,

0 | 000000000000000 | 000000000000001

In decimal, this is 2^{-16}

STORAGE FORMATS

Floating point representation: The numbers are represented by standard scientific notation, such as -49.8×10^{-3}.

In general, in **normalized scientific notation**, a finite number can be represented by four integer components:

$$(-\mathbf{1})^{s}\times\boldsymbol{m}\times\boldsymbol{b}^{e}$$

Where, **s** = sign (0 for + and 1 for -), **b** = base , **e** = exponent, where **m < |b|**.

Examples: $-4.98\times10^{-2}, 1.001\times2^{5}$

STORAGE FORMATS

- To store a floating-point binary in memory, **IEEE 754** (IEEE= Institute of Electrical and Electronics Engineers) has defined different formats:

  Single Precision that uses 32-bit of storage

  Double Precision that uses 64-bit of storage

Precision	Sign (bits)	Exponent (bits)	Significand (bits)
Single	1	8	23
Double	1	11	52

OVERFLOW AND UNDERFLOW

- The situation where an integer outside the allowed range is to be presented, which means we need more bits that can be stored, is called an **overflow**.
- With real numbers, an exponent that is too small to be stored causes an **underflow**.

ALGORITHM

An **ALGORITHM** is defined as a step-by-step procedure or method for solving a problem. Before writing a computer program (or code) we write some steps of instruction in our language independent of the computer language we use.

Example:

An algorithm to add two numbers:

1. Take two number inputs
2. Add numbers using the + operator
3. Display the result

BASIC STRUCTURE OF A COMPUTER PROGRAM

- **Sequence**

  To execute a list of statements in order:

  Example: assignment, operation, print

```
x = 2
y = 3
z = x + y
print z
```

BASIC STRUCTURE OF A COMPUTER PROGRAM

- **Repetition**

  To repeat a block of statements while a condition is true

  Example: Loop structure

  Loop starts

  Statement 1

  Statement 2

  Loop ends

BASIC STRUCTURE OF A COMPUTER PROGRAM

- **Selection**

  It is about the choice of one action from several alternative actions following some logic.

  Example:

  If (Condition)

  Statement / action

  end

MODELS OF COMPUTER PROGRAMMING

- **Structured Programming**

In structured programming (eg: C, Fortran), the computer code or program is executed one after another. The control statements (logical structures, loops etc.) dictate which block of code will be executed and what next.

- **Object Oriented Programming (OOP)**

The object-oriented programming (eg: Python) is based on the concept of 'object'. An object can be data or function or any mixed code. Data are often known as attributes and the functions are often called methods.