Problem Solving

(CS 1002)

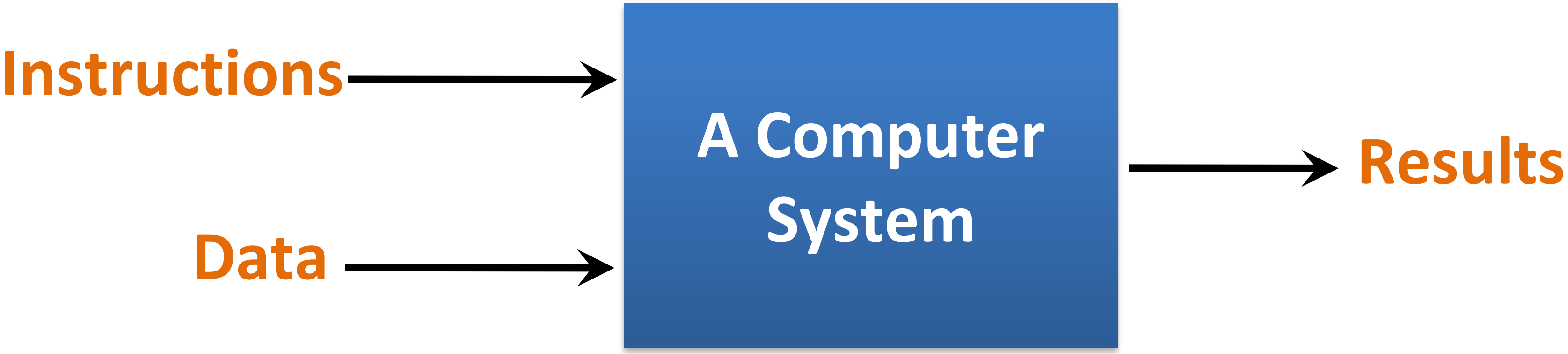
Dr. Mudassar Aslam

Cybersecurity Department

National University of Computer & Emerging Sciences,
Islamabad Campus

What is a Computer?

- A computer is a **electro-mechanical device** that works **semi-automatically** to **process input data** according to the **stored set of instructions** and produces **output** or resultant data.



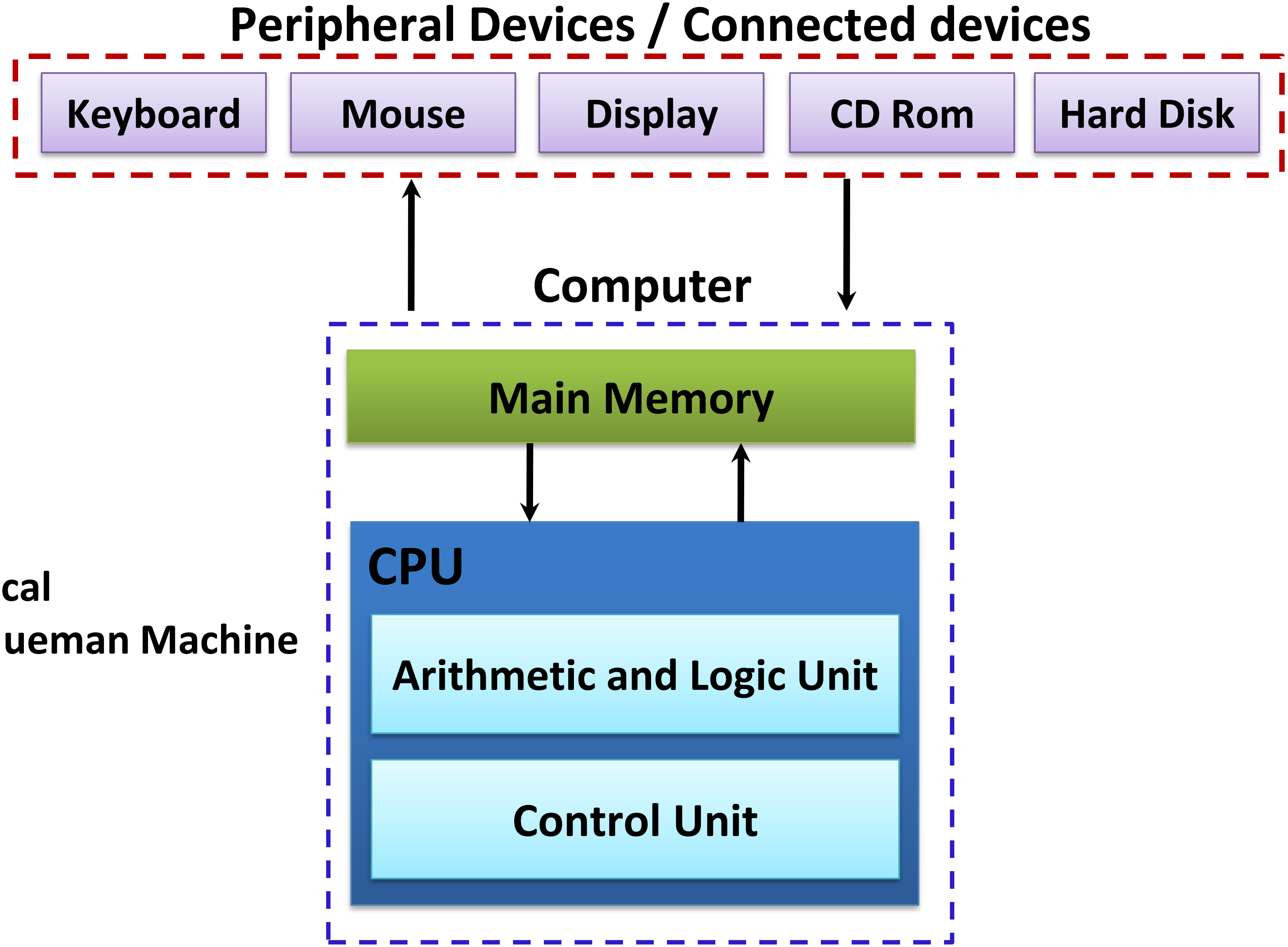
Instructions

Data

A Computer System

Results

Components of a Computer System



Peripheral Devices / Connected devices

Keyboard

Mouse

Display

CD Rom

Hard Disk

Computer

Main Memory

CPU

Arithmetic and Logic Unit

Control Unit

A Typical
Von-Nueman Machine

Computer Instructions and Programs

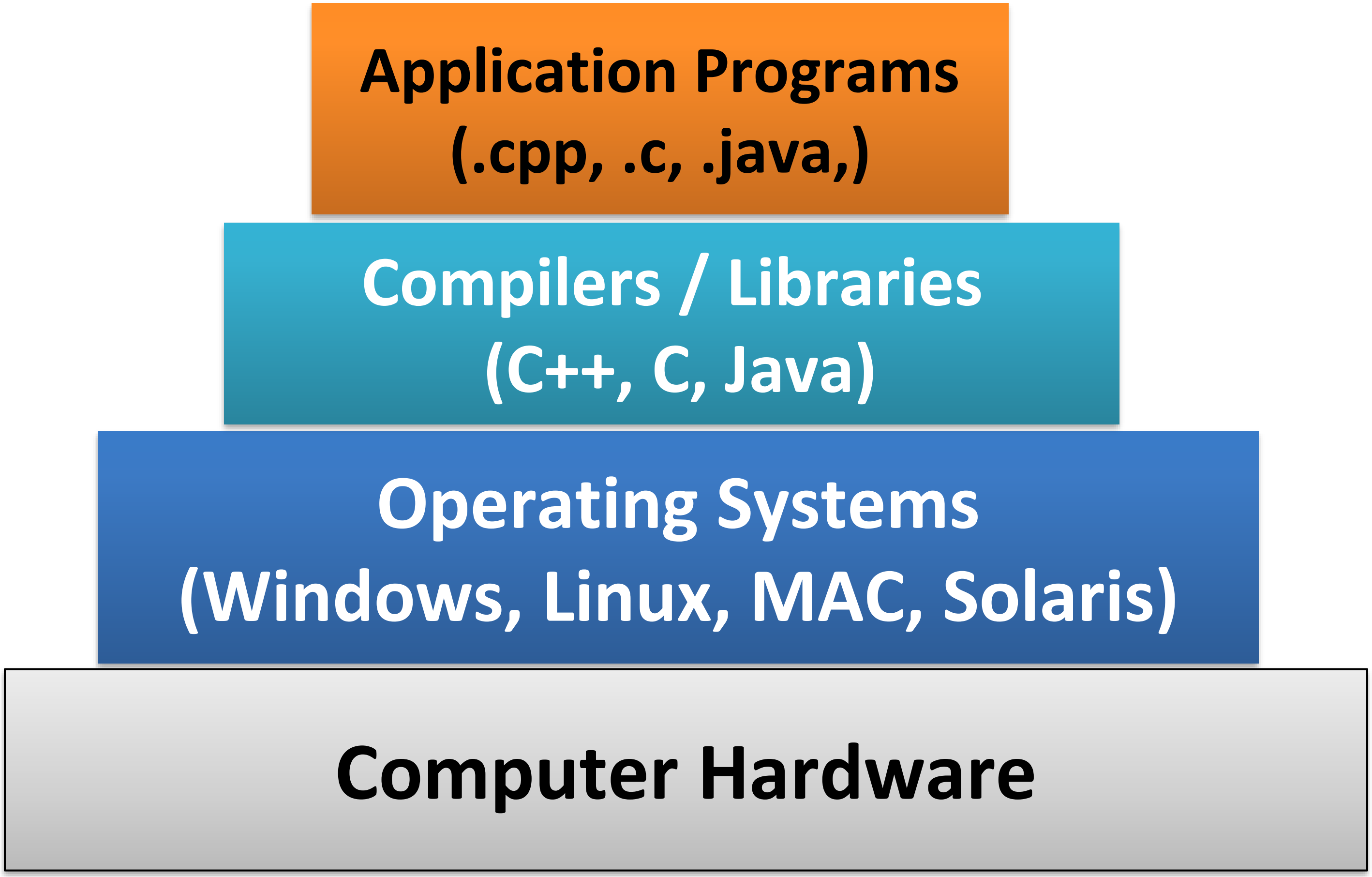
- **Instruction**: A computer instruction is a command or directive given to a computer to perform specific task.

  Examples: Add 2 and 5, Print “Hello World”

- **Program**: A **program** is **sequence** of **instructions written** in **programming language** that directs a **computer** to **solve a problem**

  Examples: Draw a square, etc.

Computer Software System



Application Programs
(.cpp, .c, .java,)

Compilers / Libraries
(C++, C, Java)

Operating Systems
(Windows, Linux, MAC, Solaris)

Computer Hardware

Programming Languages

Classification of programming languages:

1. Machine language
2. Low-level languages
3. High-level languages

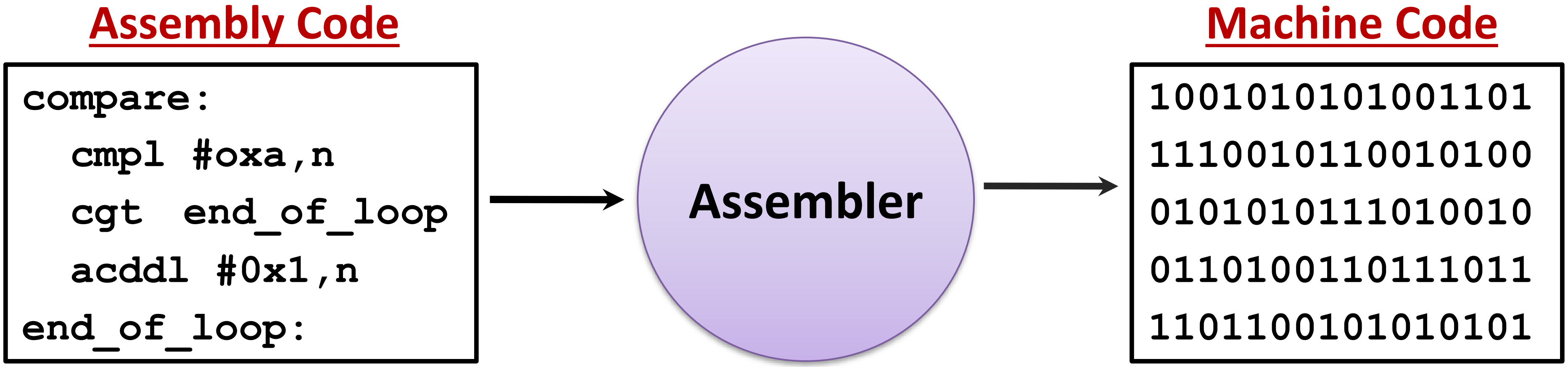
1. Machine level languages

- A **computer understands** only **sequence of bits or 1's and 0's** (the smallest piece of information)

- A **computer program** can be **written** using **machine languages** (**010011010100100010....**)
  - **Very fast execution**
  - **Very difficult** to **write** and **debug programs**
  - **Machine specific** *(different codes on different machines)*

2. Low level languages

- **English encrypted words** **<u>instead of codes</u>**
- **More understandable** (*for humans*)
- Example: **Assembly language**
- Requires: “**Translation**” from **Assembly** code to **machine code**



```
compare:
   cmpl #oxa,n
   cgt  end_of_loop
   acddl #0x1,n
end_of_loop:
```

```
1001010101001101
1110010110010100
0101010111010010
0110100110111011
1101100101010101
```

3. High level languages

- **Mostly machine independent**
- Close to **natural language** (English like keywords)
- **Easy to write** and **understand** programs
- **Easy to debug** and **maintain** code
- **Requires compilers** to **translate** to machine code
- **Slower** than **low-level languages**

3. High level languages

- Many popular High-Level languages

Java
Ruby
C#
php
Objective-C
C++
python
Perl
GO
JavaScript
THE C PROGRAMMING LANGUAGE
Visual Basic

https://c1.staticflickr.com/4/3912/15090961835_c4f26e4890_b.jpg

Problem Solving Steps

1. **Understand** the problem
2. **Plan** the logic
3. **Code** the program
4. **Test** the program
5. **Deploy** the program into production

1. Understanding the Problem

- **Problems** are often **described** in **natural language like English**.

- **Identify the requirements**

  1. **Inputs** or given **data-items**
  2. Required **output(s)** or desired results
  3. **Indirect inputs** (may not be given directly, you have to calculate or assume)

1. Understanding the Problem

- Example: **Calculate** the **area of a circle** having the radius of 3 cm
  - Inputs:

    Radius=3
  - Output:

    Area
  - Indirect Inputs:

    Pi=3.14

Area = 3.14 * (3*3) = 28.27

2. Plan the Logic

- **Identify/Outline** small **steps** in sequence, to **achieve** the **goal** (or desired results)

- Tools such as ***flowcharts*** and ***pseudocode*** can be used:
  1. **Flowchart**: a **pictorial representation** of the **logic** steps
  2. **Pseudocode**: **English-like representation** of the **logic**

***Advice**: Walk through the logic before coding*

3. Code the Program

- **Code the program:**
  - **Select** the programming **language**
  - **Write** the **program instructions** in the selected programming language
  - Use the **compiler** software to translate the program **into machine understandable code**
  - **Syntax errors** (Error in **program instructions**) are **identified** by the **compiler** during **compilation** and **can be corrected**.

4. Test the Program

- **Testing the program**
  - **Execute** using **sample data** and **check** the **results**
  - **Identify** **logic errors** if any (***undesired results or output***) and correct them

5. Deploy the Program

- **Putting the program into production**
  - Do this after **testing is complete** and all **known errors** have been **corrected**

Introduction to Pseudocode

- One of the popular representation based on natural language

- Widely used
  - Easy to read and write
  - Allow the programmer to concentrate on the logic of the problem

- Structured in English language (Syntax/grammar)

What is Pseudocode (continued...)

- English like statements
- Each instruction is written on a separate line
- Keywords and indentation are used to signify particular control structures.
- Written from top to bottom, with only one entry and one exit
- Groups of statements may be formed into modules

https://www.google.com.pk/url?sa=i&url=https%3A%2F%2Fimgflip.com%2Fi%2Fnp6r7&psig=AOvVaw0hSaZQ4QperYmr4Q-0159s&ust=1630516336101000&source=images&cd=vfe&ved=2ahUKEwijzbDp4NvyAhX3AmMBHQ7ZB7QQjRx6BAgAEAk