Programming Fundamentals

2022-batch BS (CySec)

Course teacher: Dr. Muhammad Awais,
Assistant professor, Department of Cyber Security

Course website: https://tinyurl.com/22cys-pf

Email: awaisrajput@quest.edu.pk
engr.awais.rajput@gmail.com

About the course

- 3+1 credit hours course (100+50)
- Marks distribution (see Table 1)
- Theory
  - 4 weekly lectures
- Practical lab
  - 3-hour lab
  - Two groups

Task	Marks	
	Th.	Pr.
Assignments, quizzes, tests	20	20
Mid-semester exams	20	--
Final exams:	60	10
Viva	--	20
Total	**100**	**50**

Table 1: Marks distribution for the course

About the course instructor

- B.E (CSE), M.E (CSE)
  - *QUEST Nawabshah*
- PhD (Computer Science)
  - Universität Paderborn, Germany
  - Topic of study: Approximate computing
- Research interests
  - Approximate computing
  - Computer architecture
  - Machine learning

Course objectives

- To
  - become familiar about computer programming.
  - learn the fundamentals of problem-solving ideas in computer programming.
  - be able to describe how data are represented, manipulated, stored in a computer and
  - be able to use the fundamental concepts of data types, structured programming, algorithmic design, and user interface design.

Course contents

- Problem solving
- Review of Von-Neumann architecture
- Introduction to programming
- Role of compiler and linker
- Introduction to algorithms
- Basic data types and variables
- Input/output constructs
- Arithmetic operators
- Comparison and logical operators

Course contents (contd.)

- Conditional statements and execution flow for conditional statements
- Repetitive statements and execution flow for repetitive statements
- Lists and their memory organization
- Multi-dimensional lists
- Introduction to modular programming
- Function definition and calling
- String and string operations
- Classes and objects
- File I/O operations

Chapter -1

- Problem solving
- Review of Von-Neumann architecture
- Introduction to programming
- Role of compiler and linker

Problem solving

- How do humans solve problems?
  - Hit and trial
  - Learn from examples
  - Process / recipe / algorithm
- What about computers?
  - Obey / execute the commands given
- Examples
  - Solving a mathematical / numerical problem e.g., finding whether a number is prime or to compute factorial
  - Cooking example
  - Do grocery
  - Build a house

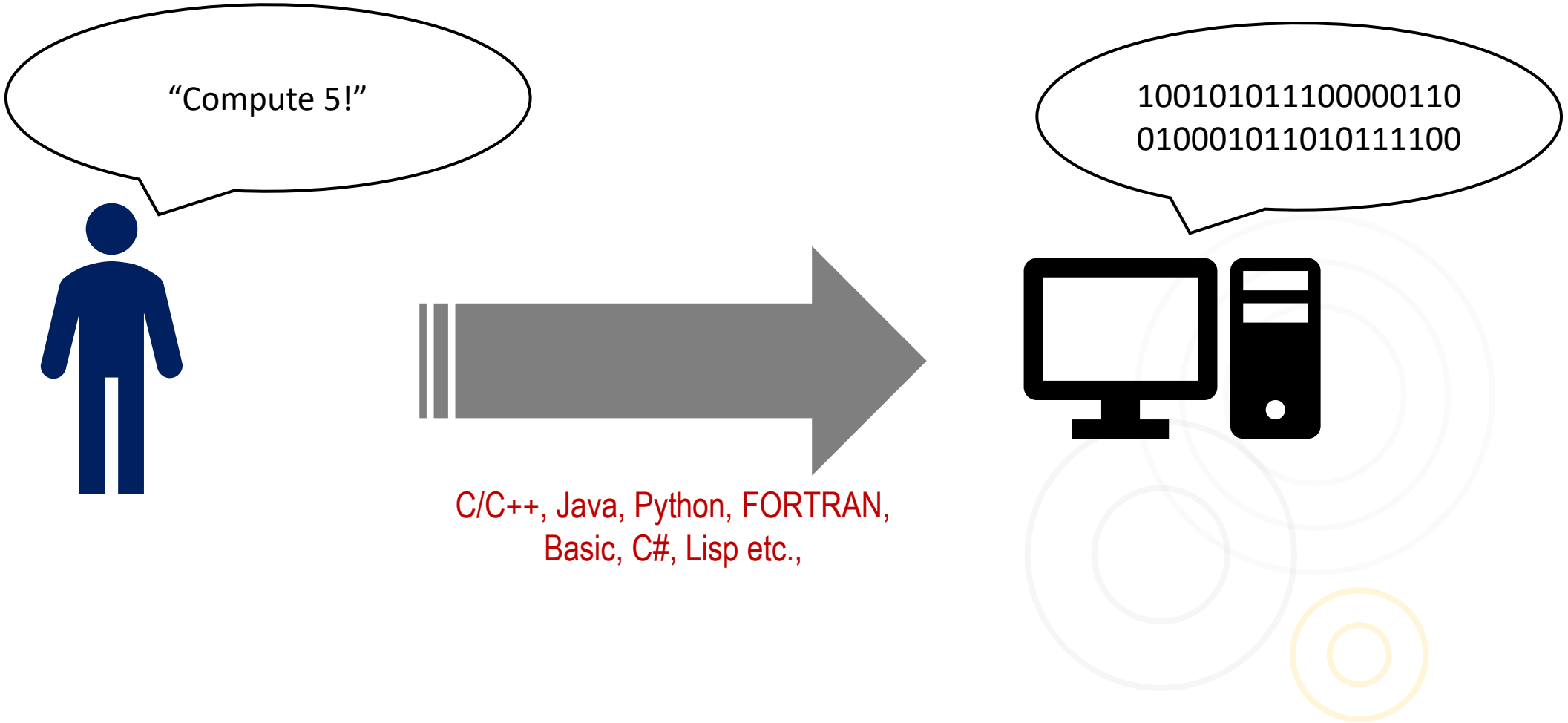
How computers are different than humans when it comes to problem solving?

Programming computers

- Computers are dumb but they can do exactly what they are told to do
- Giving instructions to a computer to perform a task is what we call “programming”
- These set of instructions are called a “computer program”
- This capability of computers is what makes them so powerful
- Example: Adding numbers from 1 to 1000
  - Program that computes the sum

How can we (humans) communicate with the machine (computer)?

Computer languages



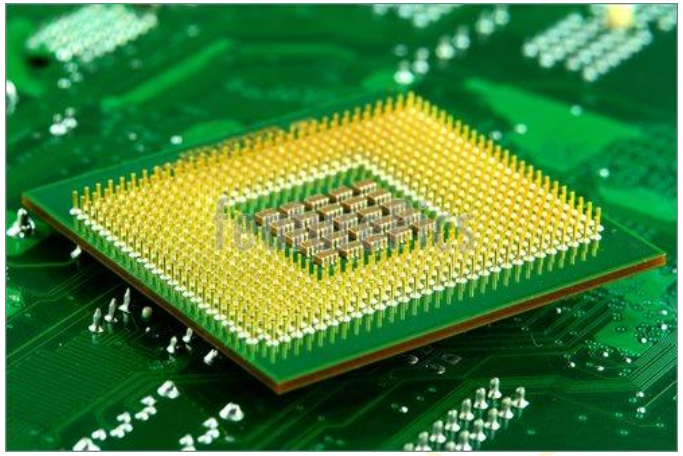
Programs

- A list of instructions given to computer to carry out a task
- Computer will execute the instructions (mostly sequentially)
- Steps to get something done with a computer program
  - Select a programming language
  - Install the compiler/interpreter
  - Write the program
  - *Compile the program
  - Run the program

```
### Checks if a node is a terminal (leaf) node
    def isTerminalNode(self,config_global):
        isTerminal=True
        total_candidates_count=0
        path_cand_count=0
        total_transformations_count=0;

        total_candidates_count = len(config_global._candidates)
        total_transformations_count = len(config_global._transformations)

        temp_node = self;

        if(temp_node._parent is None):
            path_cand_count = 0     #Special case of root node*
        else:
            while(temp_node._parent is not  None):
                path_cand_count+=1;
                temp_node = temp_node._parent
```

Microprocessor

- A micro-chip that can perform arithmetic operations
- Being a digital circuit, it understands only 0's and 1's known as "machine language"
- It can handle only simple operations like Add, Subtract, Multiply, Divide etc.
- It can be used in many devices, computer is just one example



Assembler, Linker and compiler

- It's very hard for human beings to remember different patterns of 0's and 1's
- It is almost impossible for a human being to write a computer program consisting only of 0's and 1's
- There was a need to define a language which is easier to understand and work with
- Set of short words to represent different patterns of 0's and 1's
  - Example: ADD ⇔ 00101101
- Remember: Microprocessor can only understand machine language (0s & 1s)
- How will assembly language work then?
- A translator to convert words into appropriate patterns of 0s & 1s
- i.e, converts series of instructions (written in assembly language) into machine language

Assembler, Linker and compiler (contd.)

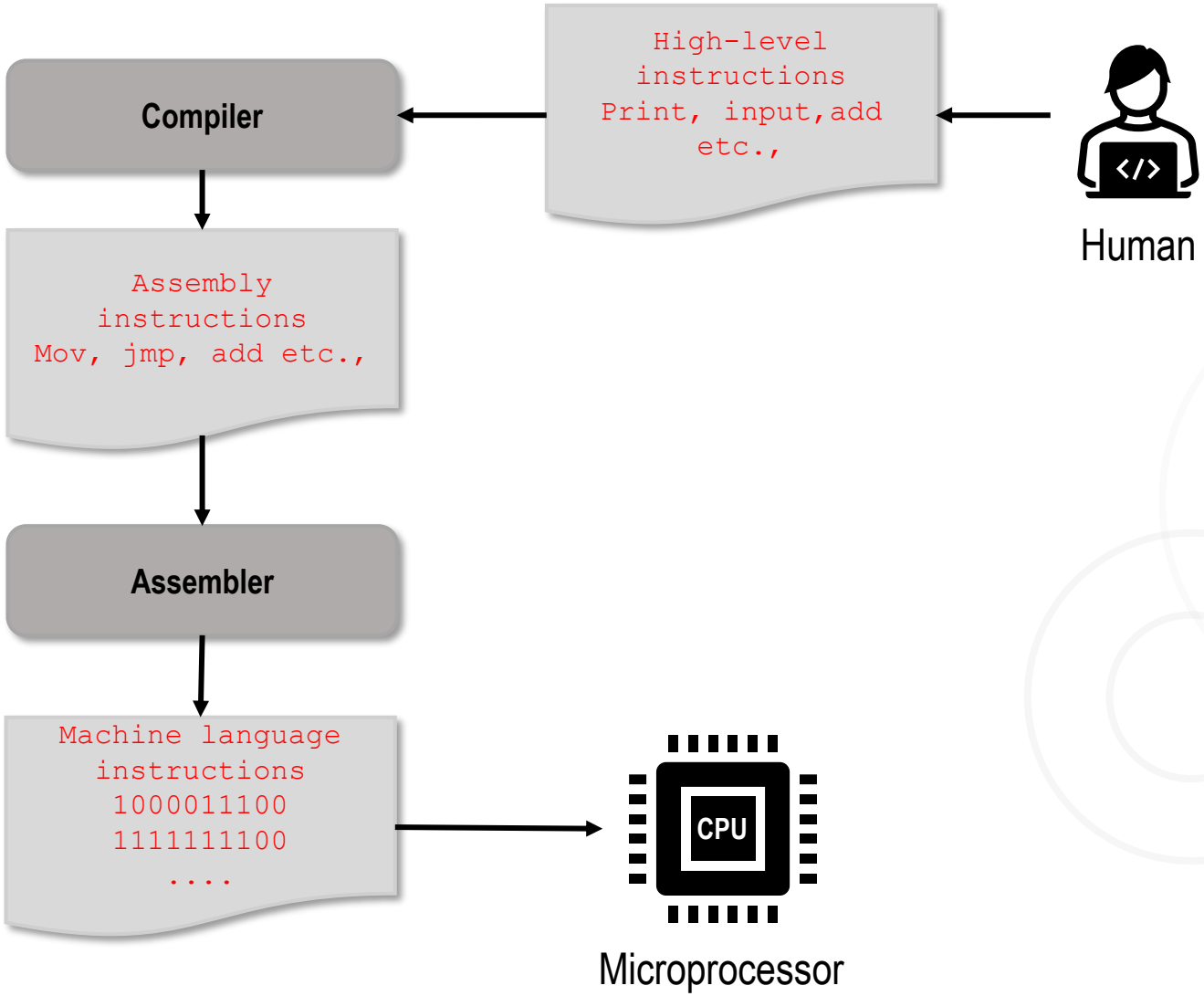
- It is closer to machine rather than human beings
- A low-level language
- Human mind does not work so simply as the machine does
- It is still very different from the languages that human beings speak
- It is still difficult for human beings to solve a complex problem by writing a computer program in assembly language
- A translator to convert instructions written in a high-level language to low-level language

A compiler converts high-level code into assembly (middle-level) language

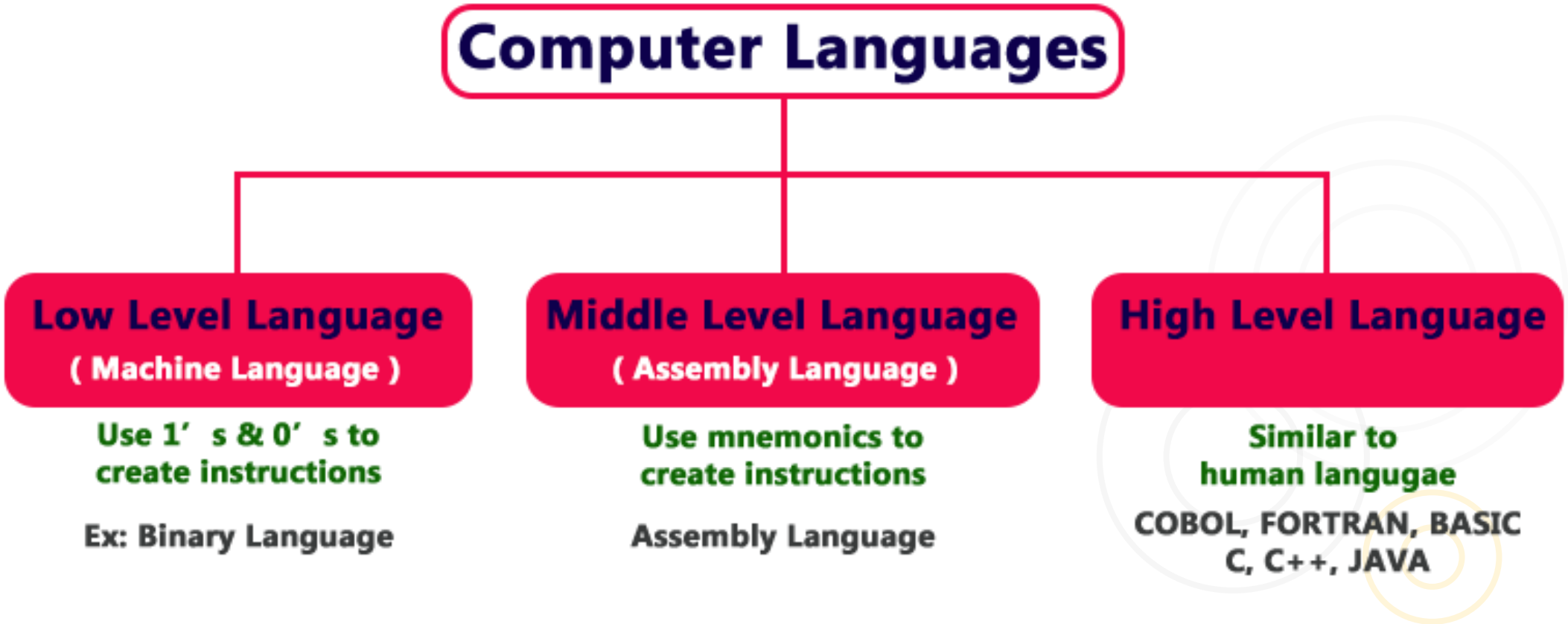
An assembler converts assembly language (middle-level) code to machine code

A linker combines different parts of code in one executable machine code

Flow of program writing



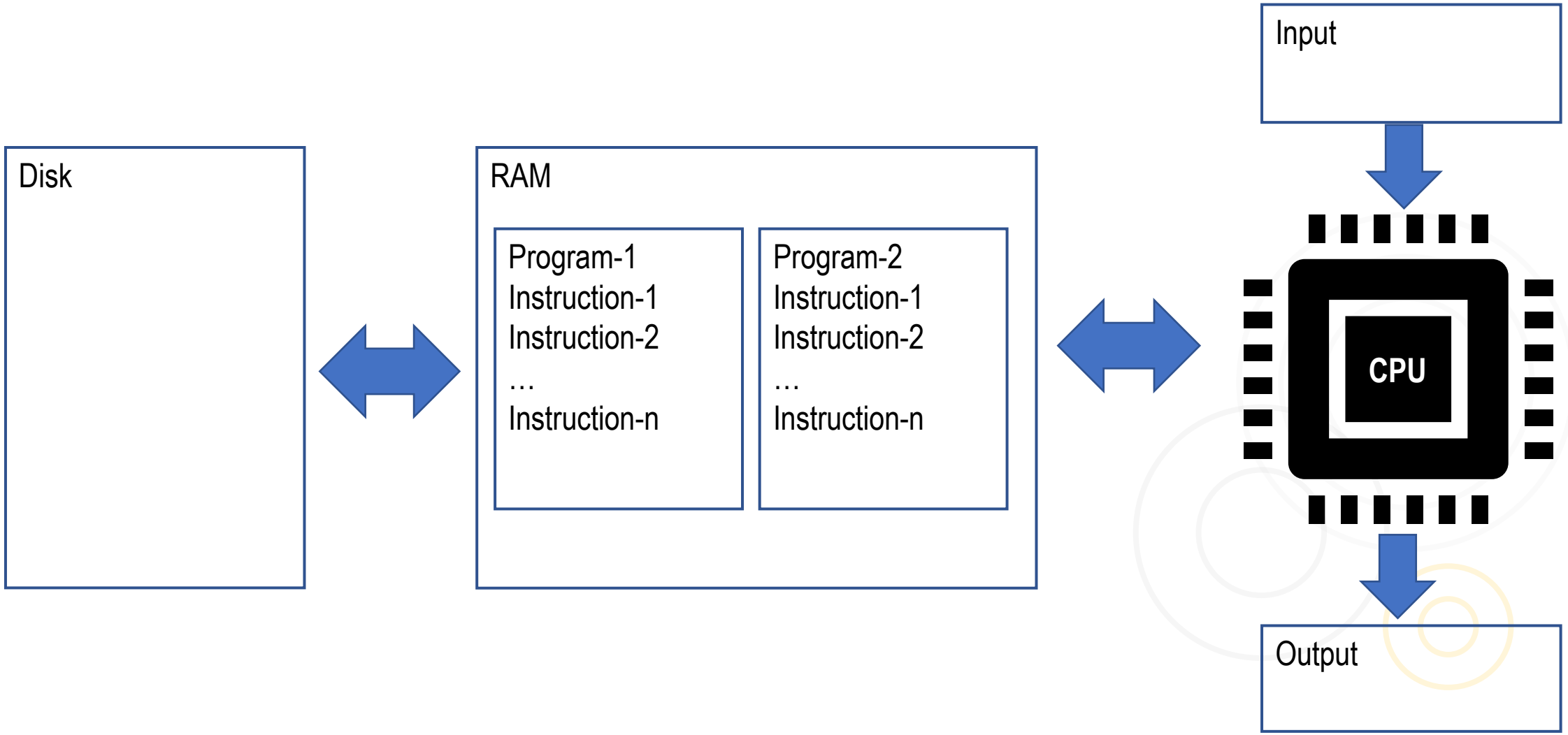
Computer languages



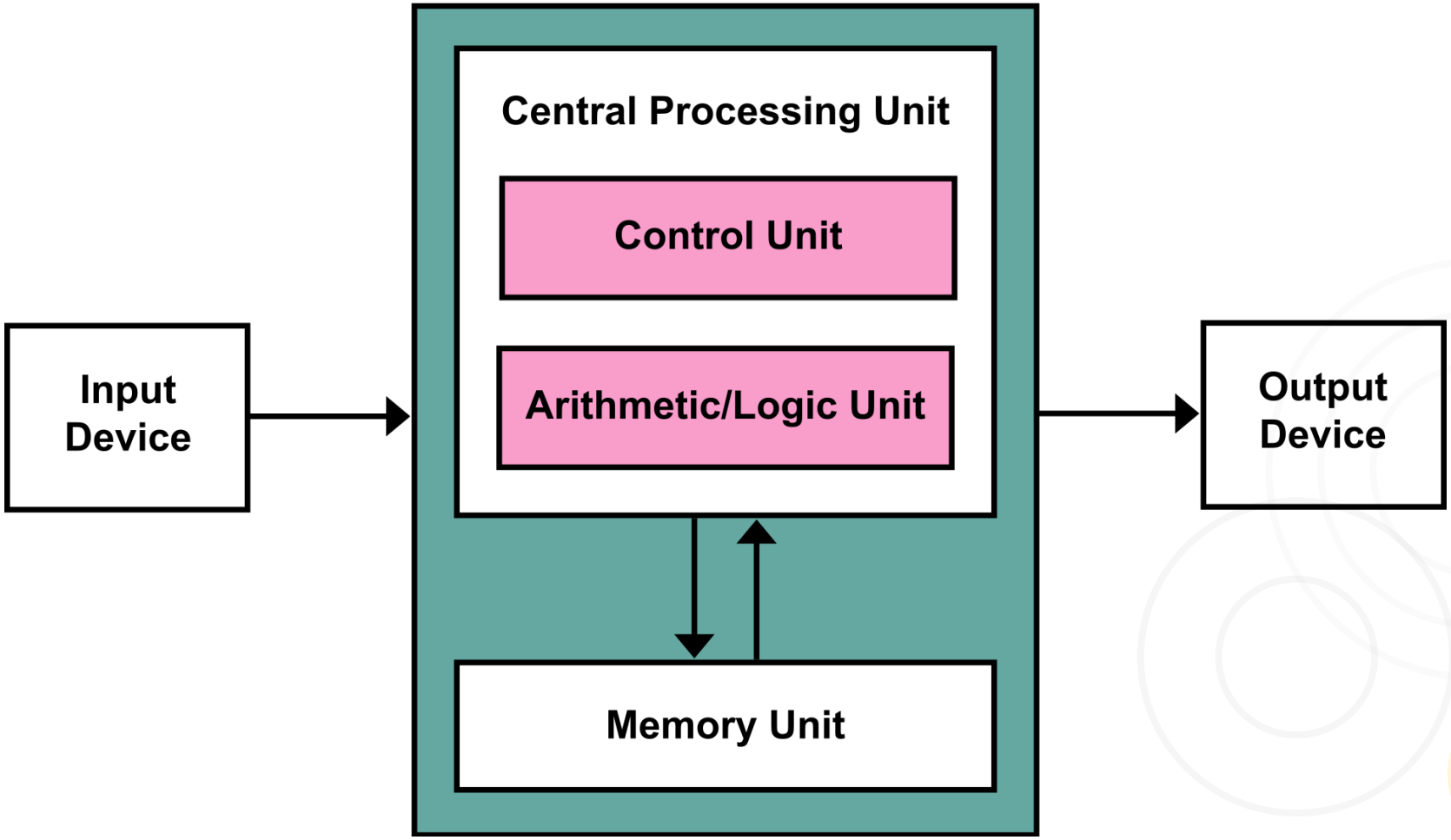
HL / LL / ML comparison

C/C++/Java:	Assembly Language	Machine Language
int X=880;	X: .word 880	
X = X + 5;	LOAD R1 X	10010011000011110000000000100000
		10011001000011110000000000000000
		01000001111100010000000000000000
	ADD R2 R1 #5	10110011000100100000000000000101
	STORE R2 X	10010011000011110000000000100000
		10011001000011110000000000000000
		01000010111100100000000000000000
	HALT	11111111111111111111111111111111

A high-level view of computer



Von Neumann architecture



https://en.wikipedia.org/wiki/Von_Neumann_architecture

References and acknowledgment

- Some of the material is taken from the slides of Dr. Umair Ali Khan
- Reference book:
  - Python crash course: a hands-on, project-based introduction to programming by Eric Matthes
- Online resources
  - https://docs.python.org/3/tutorial
  - https://www.geeksforgeeks.org
  - https://www.w3schools.com/python