Lecture Overview: Basics of Computer Architecture

- *This lecture introduces **computer architecture** as the design of computers, covering instructions, hardware components, and system organization, with historical context leading to **Von Neumann architecture**.

Key structure:

- Formal definition and two main parts (**ISA** and **HSA**)
- Distinction from implementation
- Historical background (Babbage, Lovelace, Turing, Von Neumann)
- Preview of classifications

Step 1: Formal Definition of Computer Architecture

Computer architecture is **the design of computers including their instructions, hardware components, and system organization**.

Two Main Parts

1. **Instruction Set Architecture (ISA)**
    - **Specifications** that determine **how machine language programs interact with the computer**.
    - Defines the set of instructions available for a specific computer type.
2. **Hardware System Architecture (HSA)**
    - Deals with **major hardware subsystems** like **CPU, storage, input/output**.
    - Includes **logical design** and **data flow organization** of subsystems.
    - **Determines efficiency** of the system.

Step 2: Illustration of ISA and HSA (Add 2 + 3 Example)

To understand ISA vs. HSA, consider adding **2 + 3** and storing in variable **X**.

Key Insight:

Simplification can be done in **various ways** based on available hardware.

Process Breakdown

1. **Multiple Implementation Options**
    - Different instruction sequences possible (e.g., load, add, store variations).
2. **ISA Selection**
    - **One or more ways selected** for a specific computer type—this defines the **ISA**.
    - Example: A particular instruction sequence chosen; all operations follow similar pattern.
3. **HSA Derivation**
    - Hardware chosen based on ISA.
    - For selected example: Need **at least two memory locations** and **an adder circuit**.
    - This is the **rudimentary concept of HSA**.

Step 3: Architecture vs. Implementation

Critical Distinction:

- **Architecture**: Conceptual design (ISA + HSA)—**does not define implementation**.
- **Implementation**: **Realization in hardware**, including **technology choices, speed, cost**.

Relationship:

- Both **influence each other**.
- Computers with **same ISA** run **same programs** and belong to **same family**.

> It's really important to distinguish a computer's architecture from its implementation... the implementation is the realization of computer in hardware and includes the choice of technology, speed, cost and so on whereas the

> architecture doesn't really define an implementation yet they both influence one another.

Step 4: Historical Context (Why Classifications Need History)

Purpose: Understand **nomenclatures** for computer architecture classifications.

Key Milestones

1. **Analytical Engine (Charles Babbage)**
   - **First proposed mechanical general-purpose computer**.
   - Assisted by **Lady Ada Lovelace—first to conceive machine language**.
   - **Ancestor** of modern computers.
2. **Ada Lovelace Connections**
   - Tutored by **Augustus De Morgan** (De Morgan's laws in **Boolean algebra**).
3. **Alan Turing**
   - Defined **computability** (what problems computers can solve).
   - **Father of computer science and AI**.
4. **John Von Neumann**
   - **Child prodigy** and mathematician (Princeton professor by ~30).
   - Post-WWII: Assembled teams for computation advancement.
   - **Von Neumann architecture**: The **block diagram** from previous session (CPU, memory, I/O).

> This is the person behind it [block diagram]. This architecture is named also after him and known as Von Neumann architecture.

Step 5: Summary Quote on Computing Pioneers

> Professor David Brailsford: "If Turing is the father and Babbage was the grandfather and if Lady Ada countess of Lovelace was the great aunt then John Von Neumann was the impossibly talented, impossibly charismatic, very wealthy Uncle to Computing."

Lecture Conclusion: Ready for **classifications of computer architecture** in next session.