جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

College of Sciences
Intelligent Medical System Department

Lecture 8

Instruction Set Architecture and Design

Subject: Computer Fundamentals

Level: First

Lecturer: Asst. Lect. Ali Saleem Haleem

Introduction

Microarchitecture and Instruction Set Architecture (ISA) are two fundamental concepts in computer organization. When we use a computer or smartphone, there's a lot going on behind the scenes in the processor (CPU). Two important parts that make everything work are:

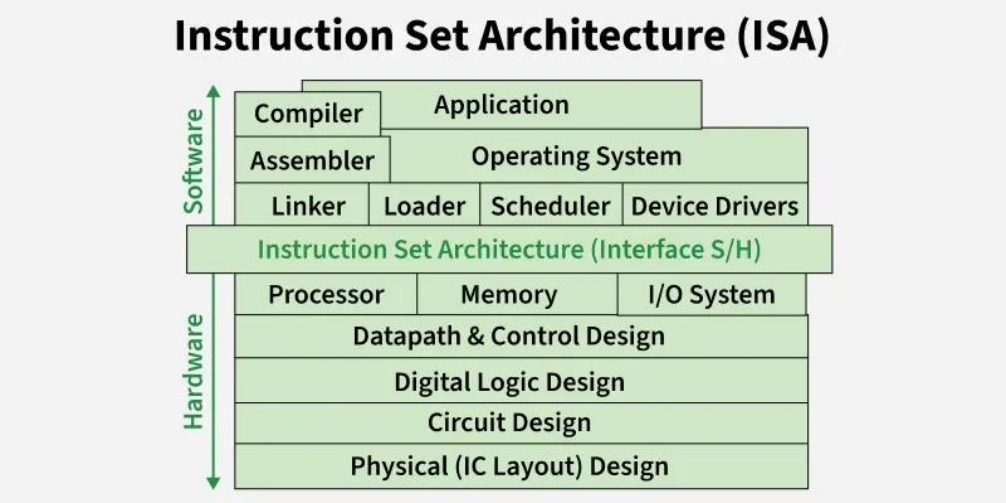
- **I**SA (Instruction Set Architecture)
- Microarchitecture

Let's understand what they mean, how they're different, and how they work together — in simple terms.

Instruction Set Architecture (ISA)

ISA is the language of the CPU that tells it what operations it can perform, such as adding numbers, loading data, or jumping to another instruction.
It defines how software communicates with hardware through specific instruction rules and formats. It includes:

- Instruction types (ADD, LOAD, JUMP), registers, data types, and memory access
- Interrupt handling and system-level communication



Objective of ISA - MIPS ISA

To understand what an ISA aims to do, let's take MIPS ISA as an example. MIPS is popular in computer science courses because it's simple and clean.

Defines Types of Instructions

MIPS divides instructions into three main types:

- Arithmetic/Logic Instructions perform basic operations such as ADD, SUB, AND, and OR on data stored in registers.
- Data Transfer Instructions are used to move data between memory and registers; for example, LW (load word) and SW (store word).
- Branch and Jump Instructions control the execution flow of the program, making decisions and handling loops or function calls; examples include BEQ (branch if equal) and J (jump).

Defines Instruction Length

MIPS is a 32-bit ISA, meaning every instruction must be exactly 32 bits (4 bytes) long. This fixed length simplifies the design and makes it more efficient for both hardware and compiler developers.

Defines Instruction Formats

Since all MIPS instructions are 32 bits long, the ISA defines how those 32 bits are organized for different instruction types. MIPS uses three instruction formats:

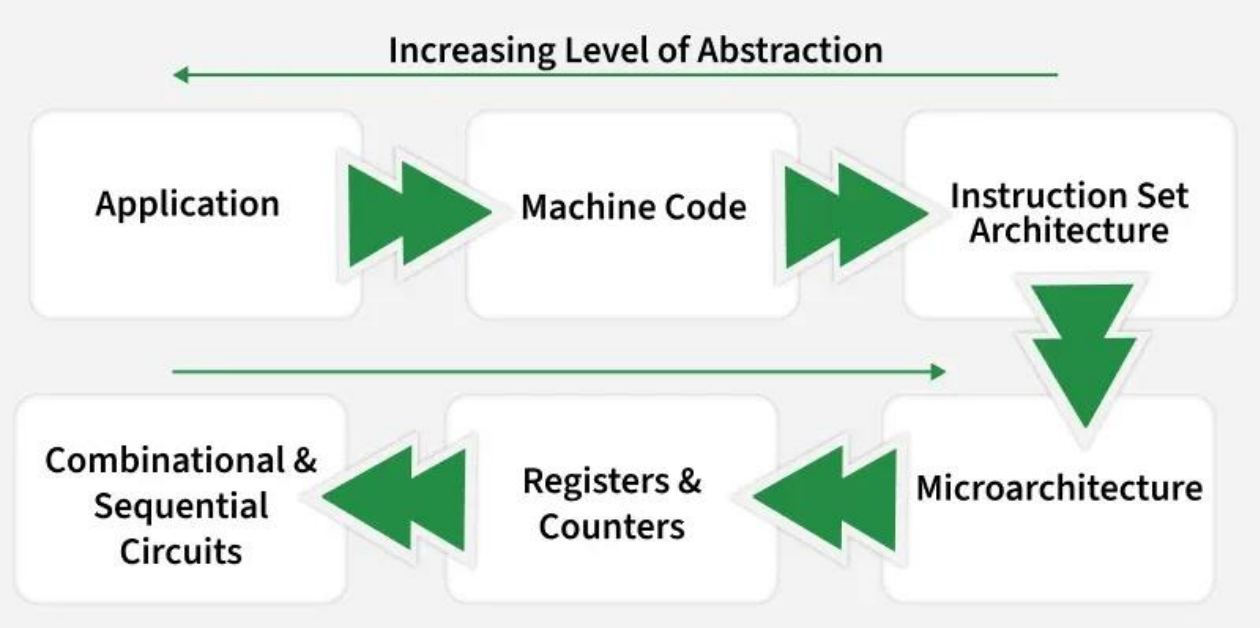
Format	Used For
R-type	Arithmetic and logic operations (eg; ADD, SUB)

Format	Used For
I-type	Data transfer and conditional branches (eg; LW, BEQ)
J-type	Unconditional jumps (eg; J)

Microarchitecture vs. ISA

Microarchitecture includes components like the ALU for calculations, pipelines for faster processing, cache for quick memory access, the control unit, and execution units.

- Processors with the same ISA, can have very different microarchitectures.
- ISA defines what a CPU can do, while microarchitecture is how the CPU is designed internally to carry out those instructions.



Importance of ISA

1. Foundation of Processor Design

ISA forms the core design element of any processor. Whether it's RISC (Reduced Instruction Set Computing) or CISC (Complex Instruction Set Computing), the choice of ISA impacts all other design decisions.

2. Instruction Execution Understanding

Computer architecture often focus on instruction execution, pipelining, control unit design, and instruction formats — all of which are defined by the ISA.

3. Enables Assembly Language Programming

Understanding ISA is critical for **assembly-level programming**. It helps in:

- Writing instruction sequences
- Understanding how data is loaded/stored
- Analyzing program execution time

4. Impact on Performance Metrics

A well-designed ISA can lead to efficient hardware implementation and optimized software execution. ISA affects:

- CPI (Cycles Per Instruction)
- Instruction count
- Execution time

5. Compatibility and Portability

ISA determines software compatibility. If two processors implement the same ISA, they can run the same programs — even if their internal microarchitectures are different.

Types of ISA

There are multiple types of ISA, each designed with different goals in mind, such as simplifying instruction sets for faster execution, supporting complex operations with fewer instructions, or enabling parallel processing to improve performance.

Type	Description
RISC	Few, simple instructions for speed
CISC	Many complex instructions
VLIW	Runs multiple operations in one instruction
EPIC	Tries to run things in parallel
Stack-based	Uses a stack instead of registers