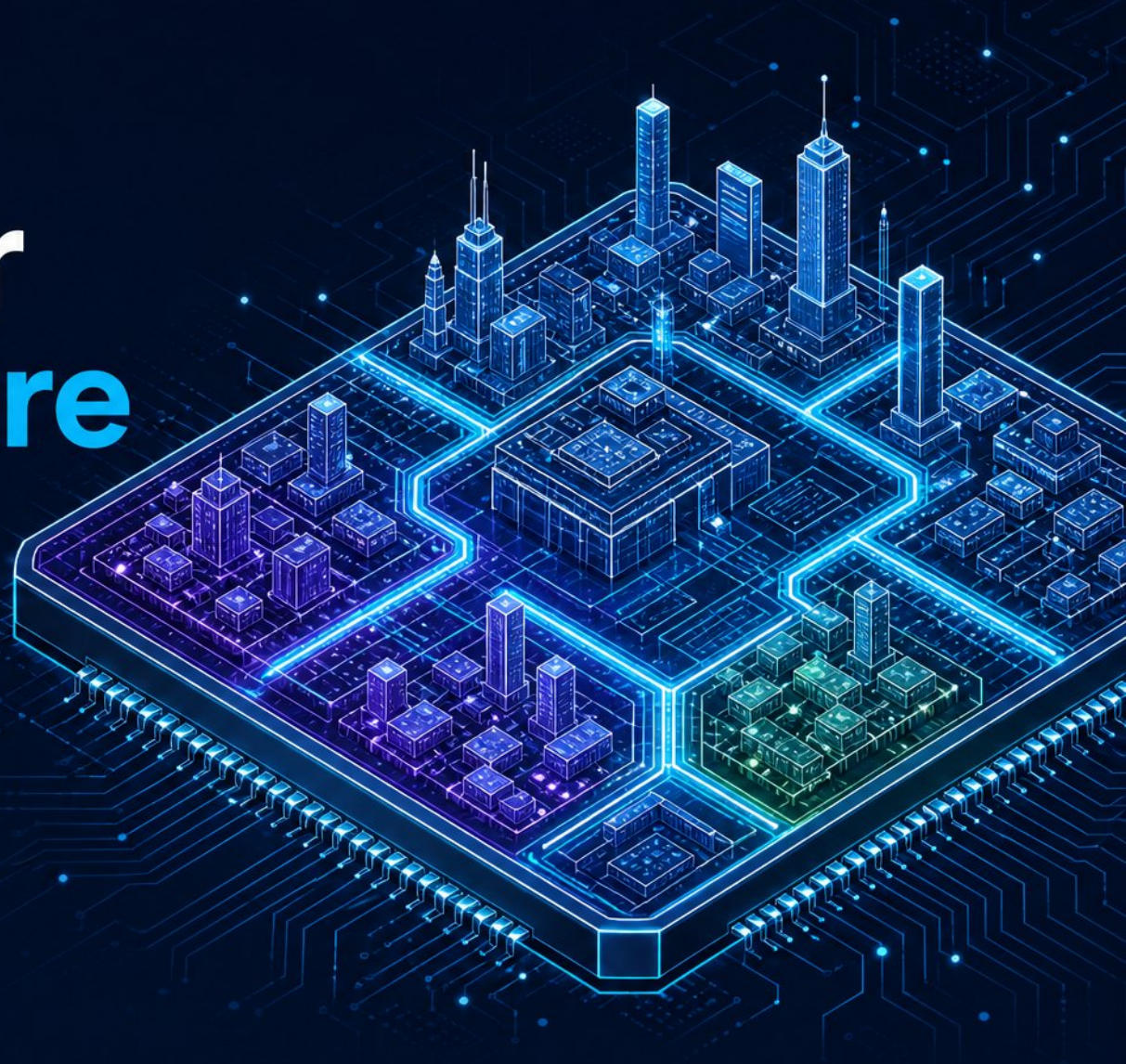
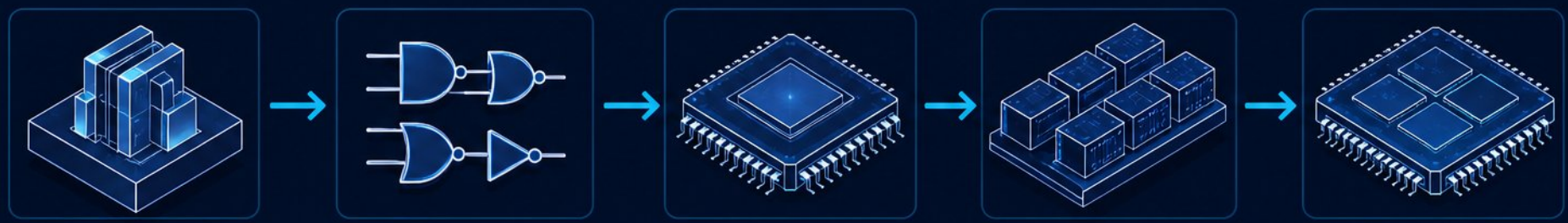
Introduction to Computer Architecture

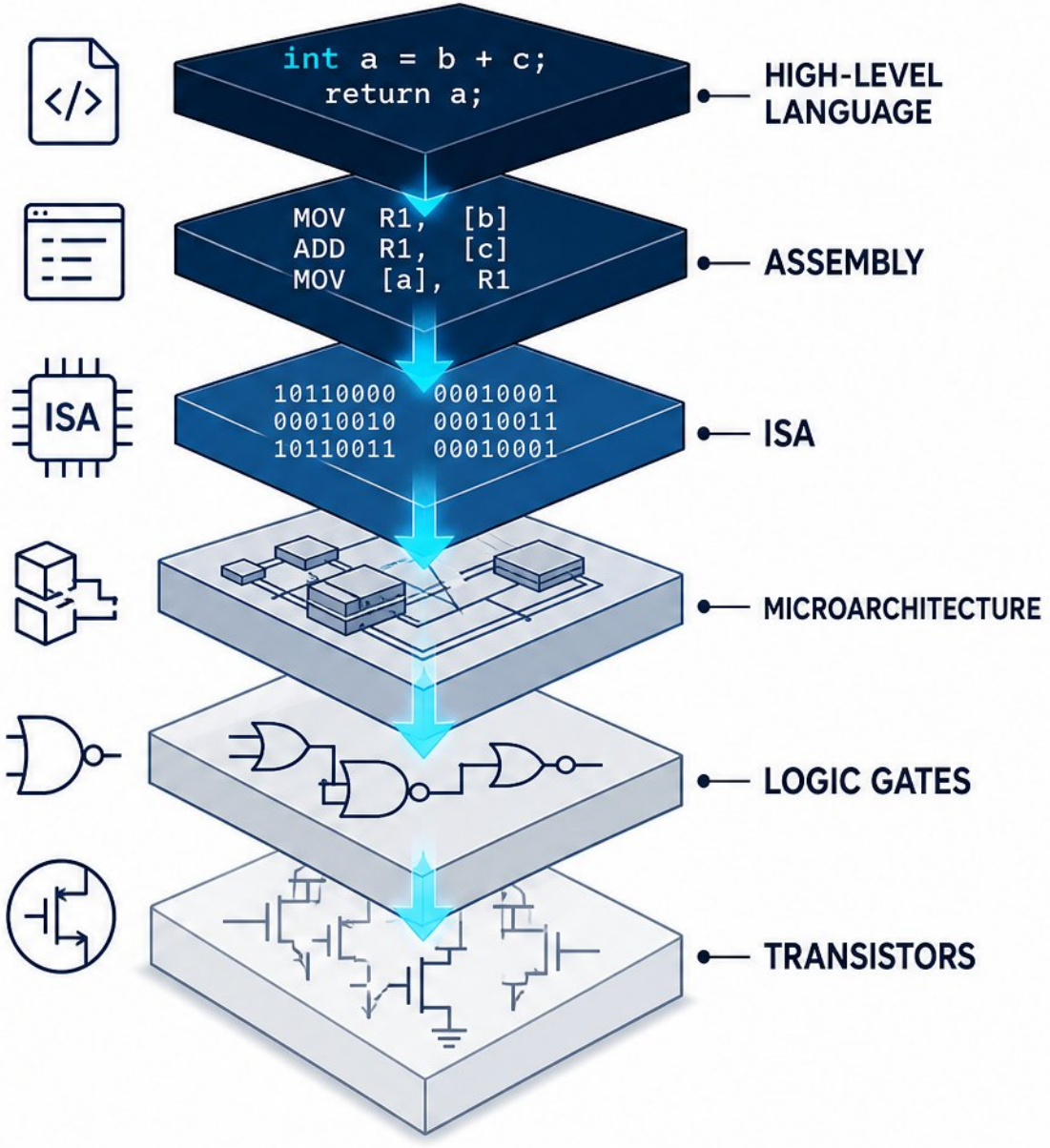


- - Bridge software behavior to the hardware executing it
- - Impacts performance debugging, cost, and security daily
- - Journey: transistors → logic → processors → memory → SoC
- - Target: beginners learning to reason about performance through a hardware lens



The Big Picture: From High-Level Code to Electrical Signals

- Source code translates down through compiler, assembly, and machine code to reach the hardware.
- Abstraction layers: High-Level Language → Assembly → ISA → Microarchitecture → Logic Gates → Transistors.
- Tracing `a = b + c` reveals the journey from a simple operation to physical electrical signals.
- ISA defines the software-visible contract; microarchitecture is the hardware implementation detail.

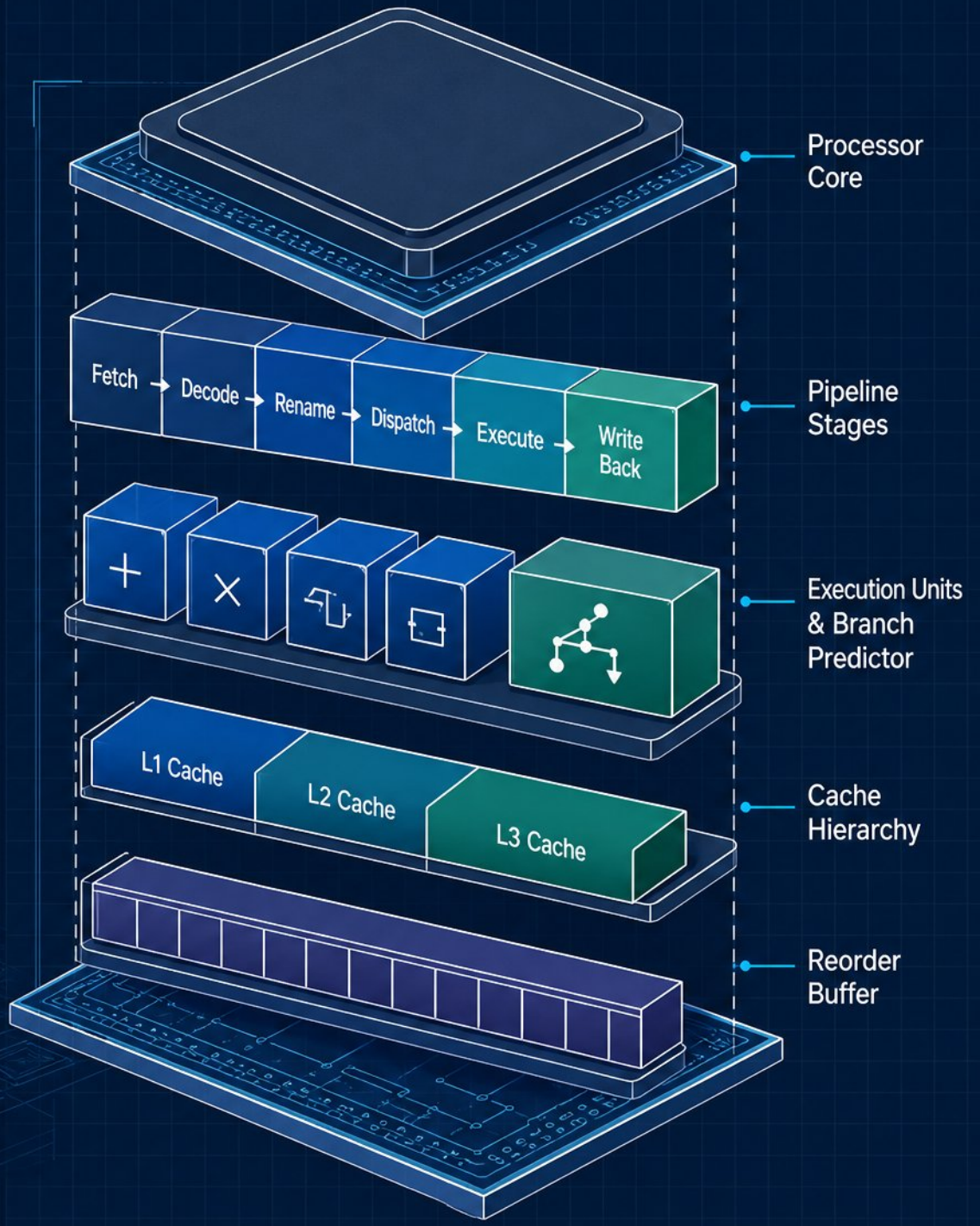


The Programmer's Contract:
Instruction Set Architecture (ISA)

- ISA defines the software-visible contract: instructions, registers, memory addressing, data types
- x86-64 (CISC), ARM, and RISC-V (RISC) – different design philosophies, same fundamental role
- Binary compatibility: same ISA means programs run on different processor implementations
- Memory model and interrupt handling are part of the ISA – critical for system-level code

Inside the Processor:
The Microarchitecture

- Microarchitecture implements the ISA: how hardware executes the software contract.
- Key components: pipeline stages, execution units, cache hierarchy, branch predictor, reorder buffer.
- Same ISA, different processors: Intel Skylake vs. AMD Zen share x86-64 but differ internally.
- Performance impact: instruction-level parallelism, speculative execution, and memory access patterns.



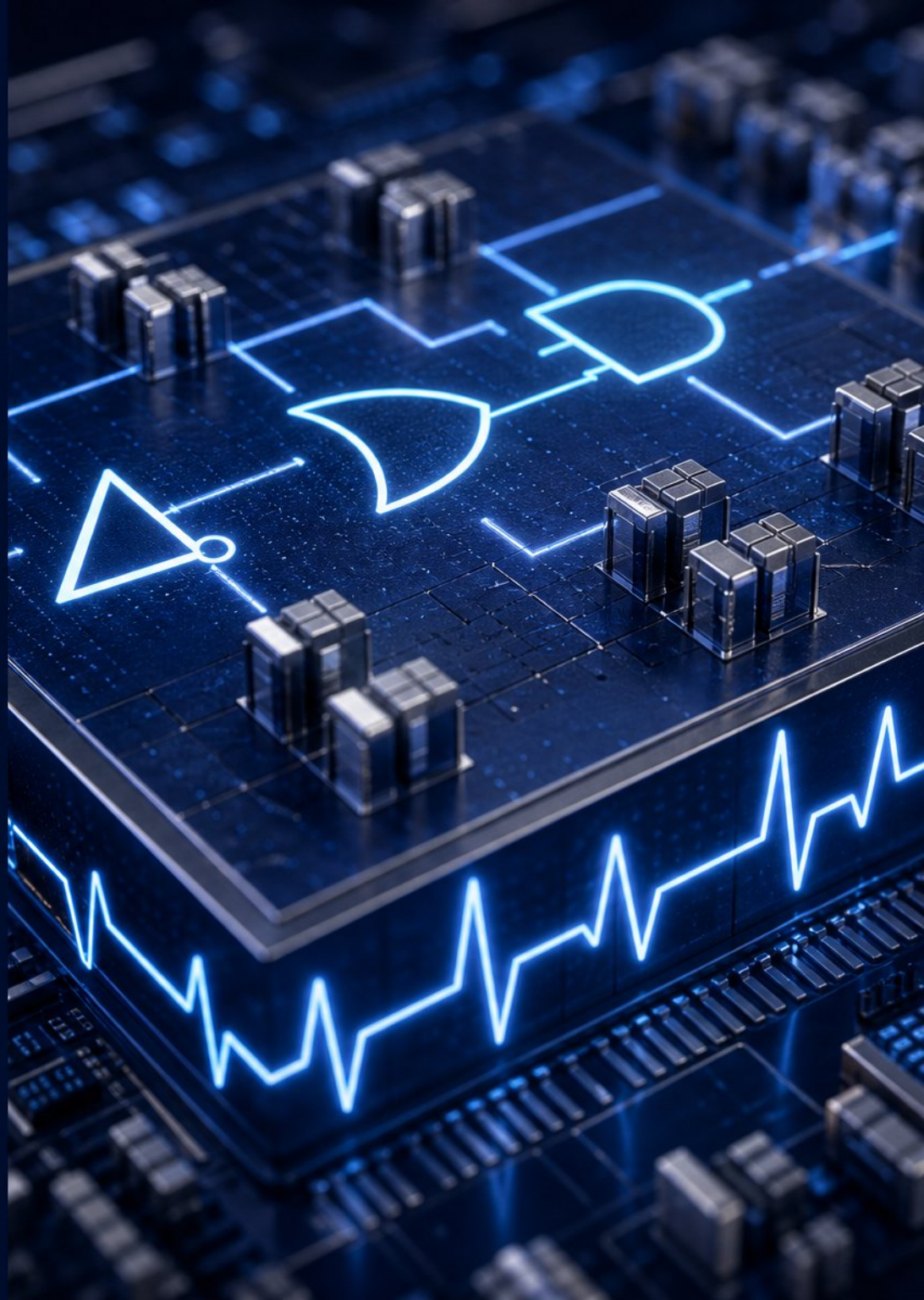
Building Blocks: Transistors, Logic Gates, and Sequential Logic

- Transistor as a switch: on/off states enable all digital computation

- Logic gates (AND, OR, NOT) combine to form adders and multiplexers

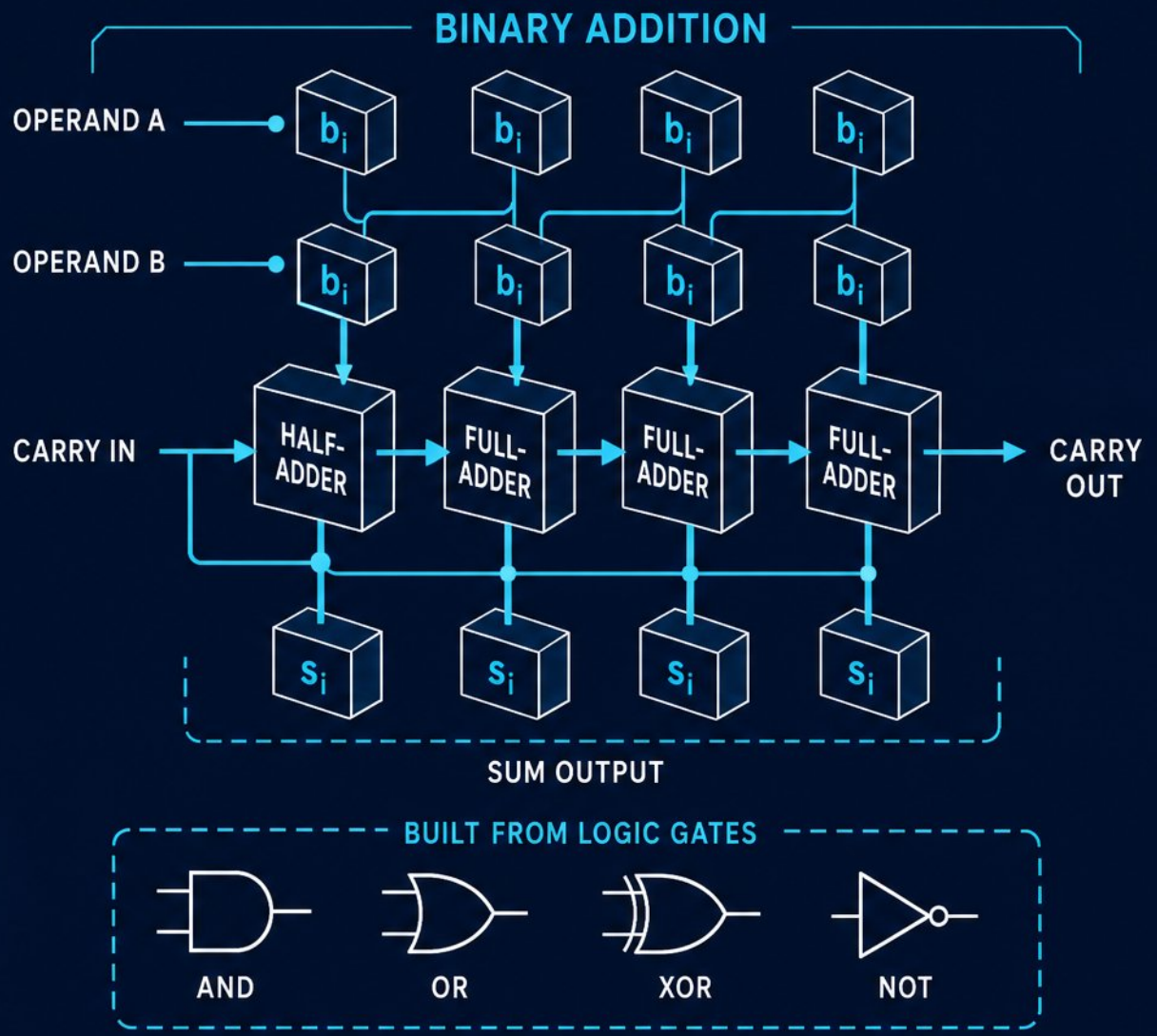
- Combinational logic computes; sequential logic stores state in flip-flops

- The clock signal synchronizes changes and sets the processor's tempo



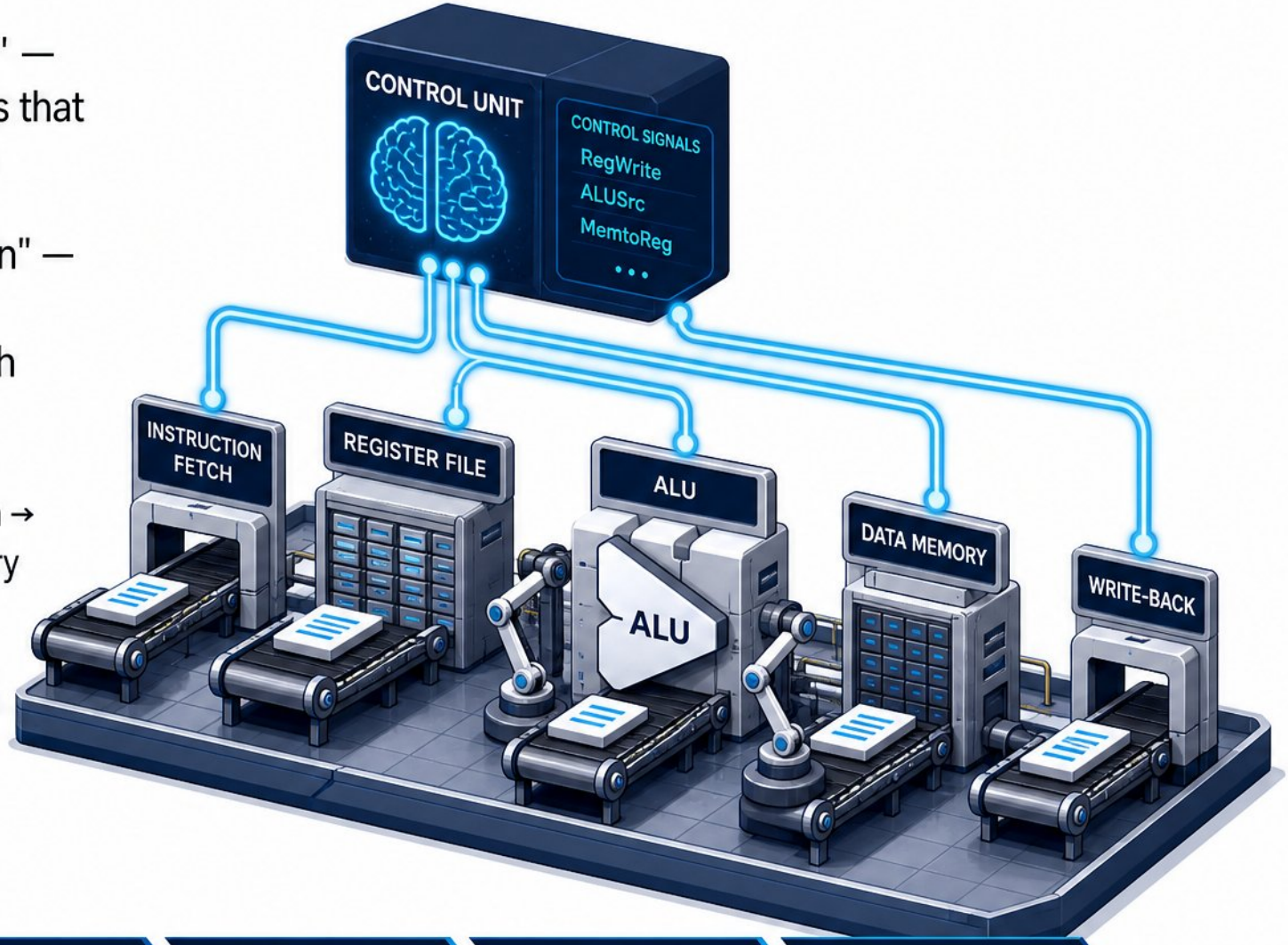
How Computers Calculate:
The Arithmetic Logic Unit (ALU)

- Core ALU operations: arithmetic, logic, and shift
- Half-adder and full-adder circuits built from logic gates
- Subtraction and comparison derived from addition
- Condition flags (zero, carry, overflow) drive if/else and loops



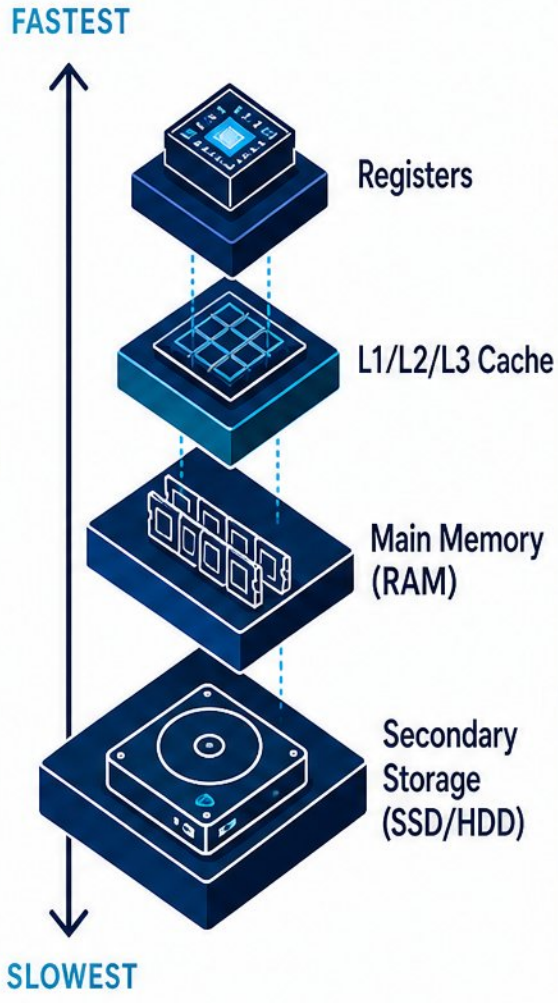
The Processor Core: Datapath and Control

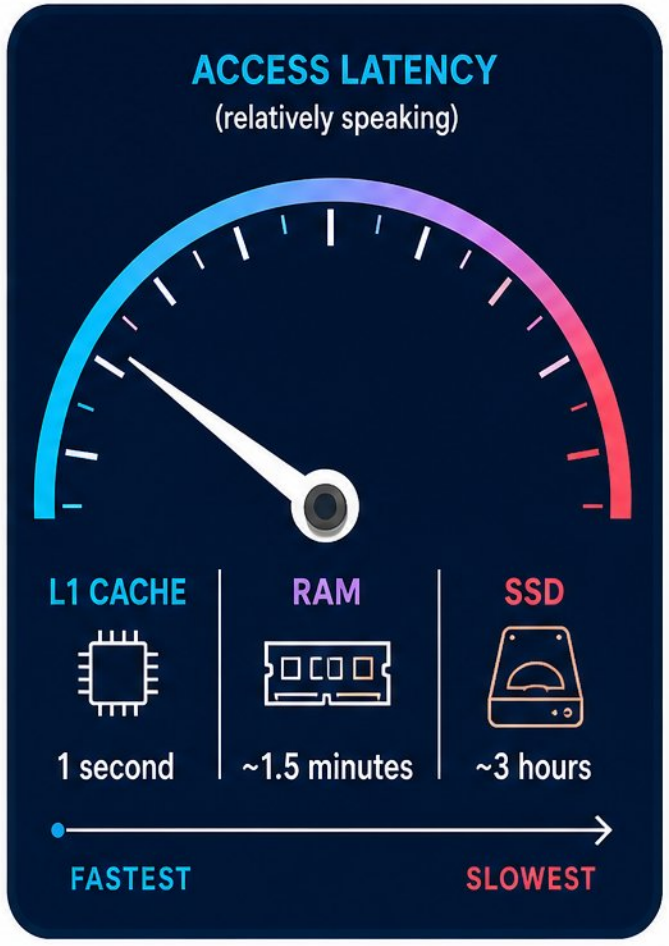
- **Datapath:** the "brawn" — ALU, registers, and buses that move and compute data
- **Control Unit:** the "brain" — decodes opcodes and orchestrates the datapath with signals
- **The classic cycle:** Fetch → Decode → Execute → Memory Access → Write-Back
- The **Control Unit** generates precise signals (RegWrite, ALUSrc, MemtoReg) to configure data flow



The Memory Hierarchy: Why Your Laptop Has Cache, RAM, and SSD

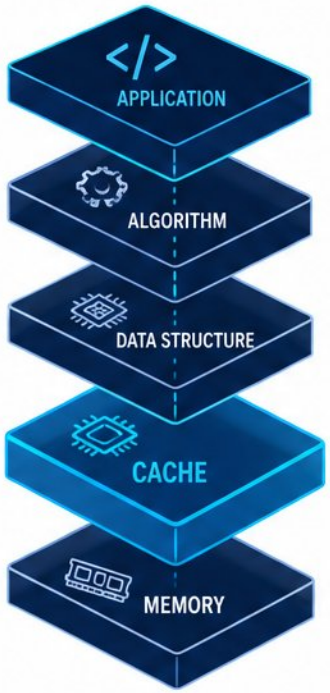
- **The "Memory Wall":** massive speed gap between processor and main memory drives the need for a hierarchy.
- **Hierarchy layers:** Registers, L1/L2/L3 Cache, Main Memory (RAM), and Secondary Storage (SSD/HDD).
- **The 100x Rule:** If L1 cache access is 1 second, RAM is ~1.5 minutes, and SSD is ~3 hours.
- **Principles of locality:** temporal and spatial locality allow caches to exploit program behavior for massive speedups.

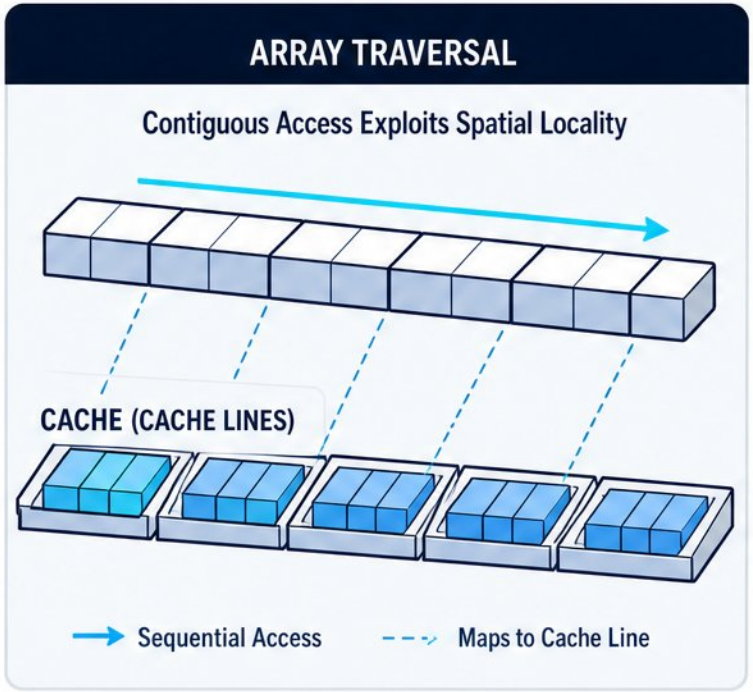


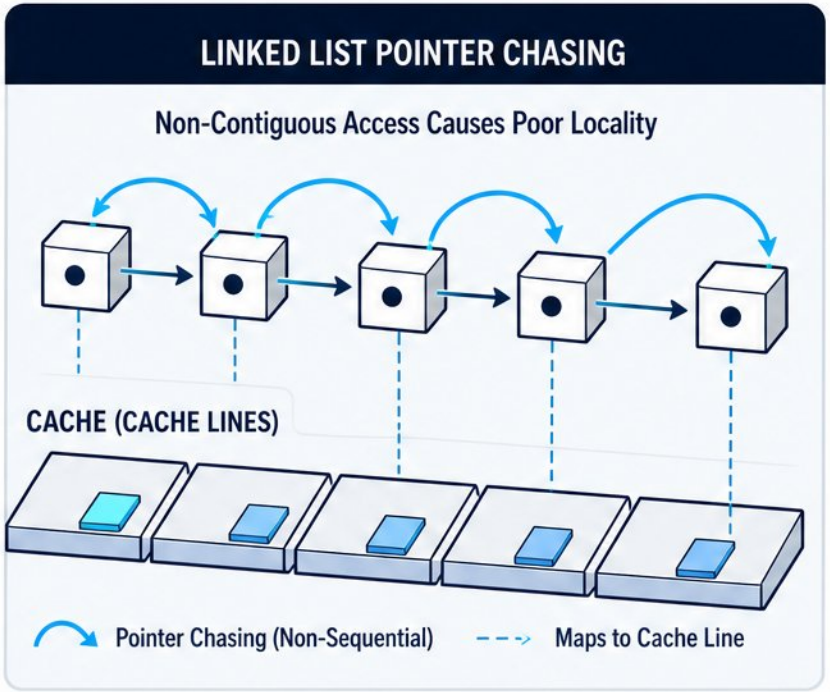


Cache Mechanics and Their Impact on Your Code

- Cache line: fundamental 64-byte transfer unit between memory and cache
- Direct-mapped vs. set-associative: hit, miss, eviction trade-offs
- Array traversal beats linked lists: contiguous access exploits spatial locality
- False sharing: threads on same cache line cause 10-100x slowdown
- L1 hit ~1s, RAM access ~1.5 min: data layout drives performance

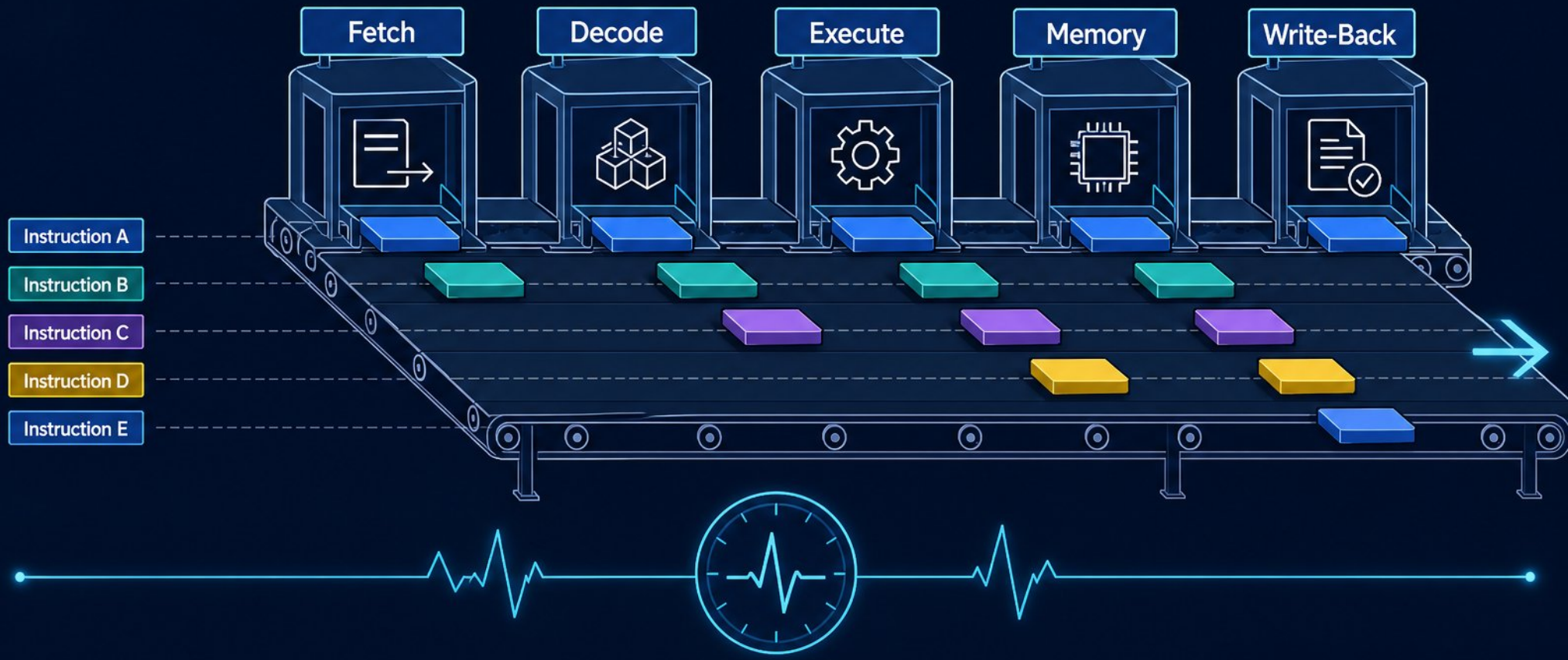






Pipelining: The Assembly Line for Instructions

- Assembly line vs. single craftsperson: overlapping instruction stages boosts throughput
- Classic 5-stage pipeline: Fetch, Decode, Execute, Memory, Write-Back
- Pipelining improves instructions per second, not single-instruction latency
- Pipeline hazards: structural, data (RAW), and control (branch mispredictions) cause stalls

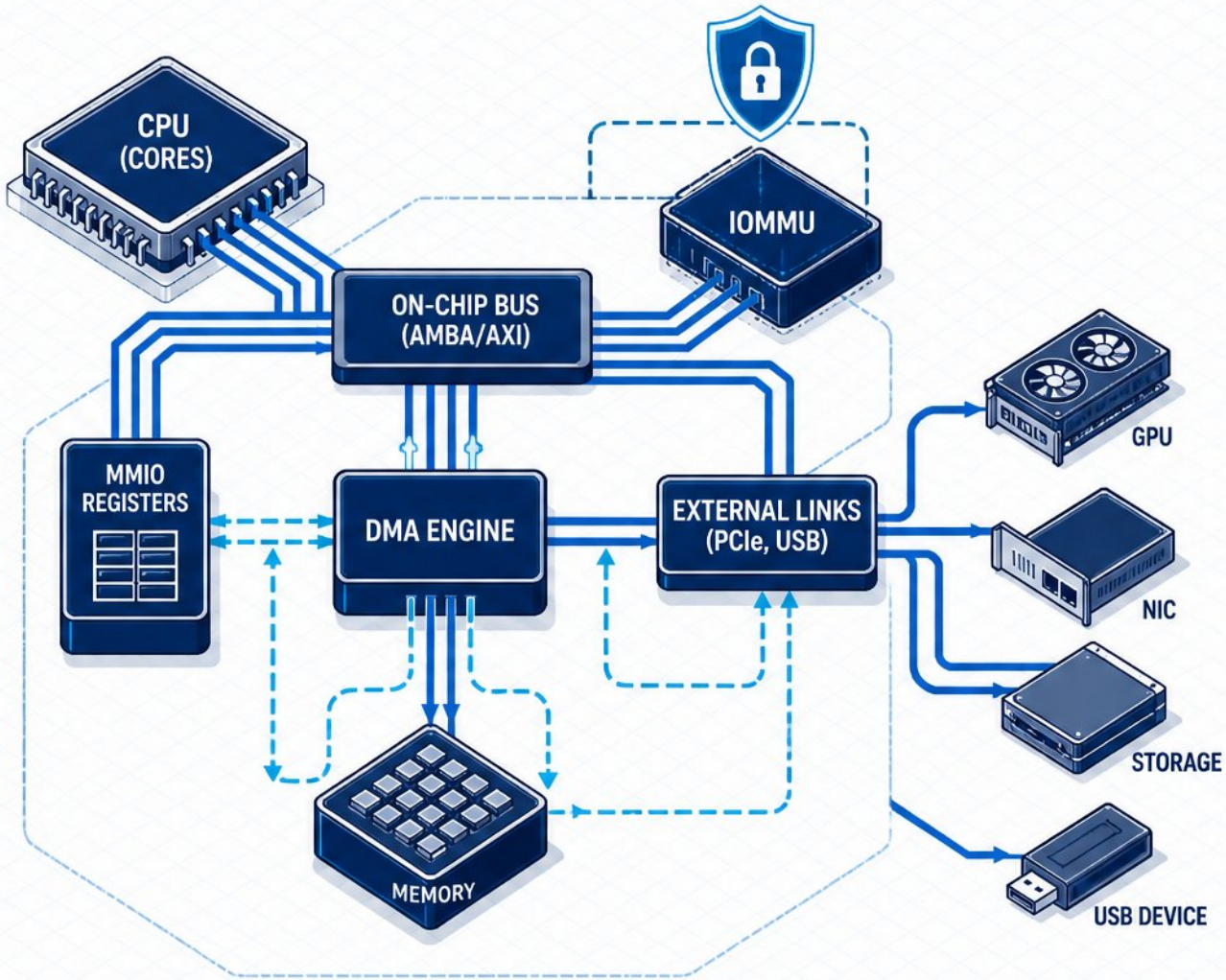


Advanced Performance: Out-of-Order and Speculative Execution

- OoO execution runs ready instructions while others wait, hiding memory latency
- Speculation guesses branch outcomes, executing ahead before the branch resolves
- Register renaming and Reorder Buffer (ROB) preserve in-order commit and correctness
- Spectre/Meltdown: speculation leaves microarchitectural traces that leak secrets

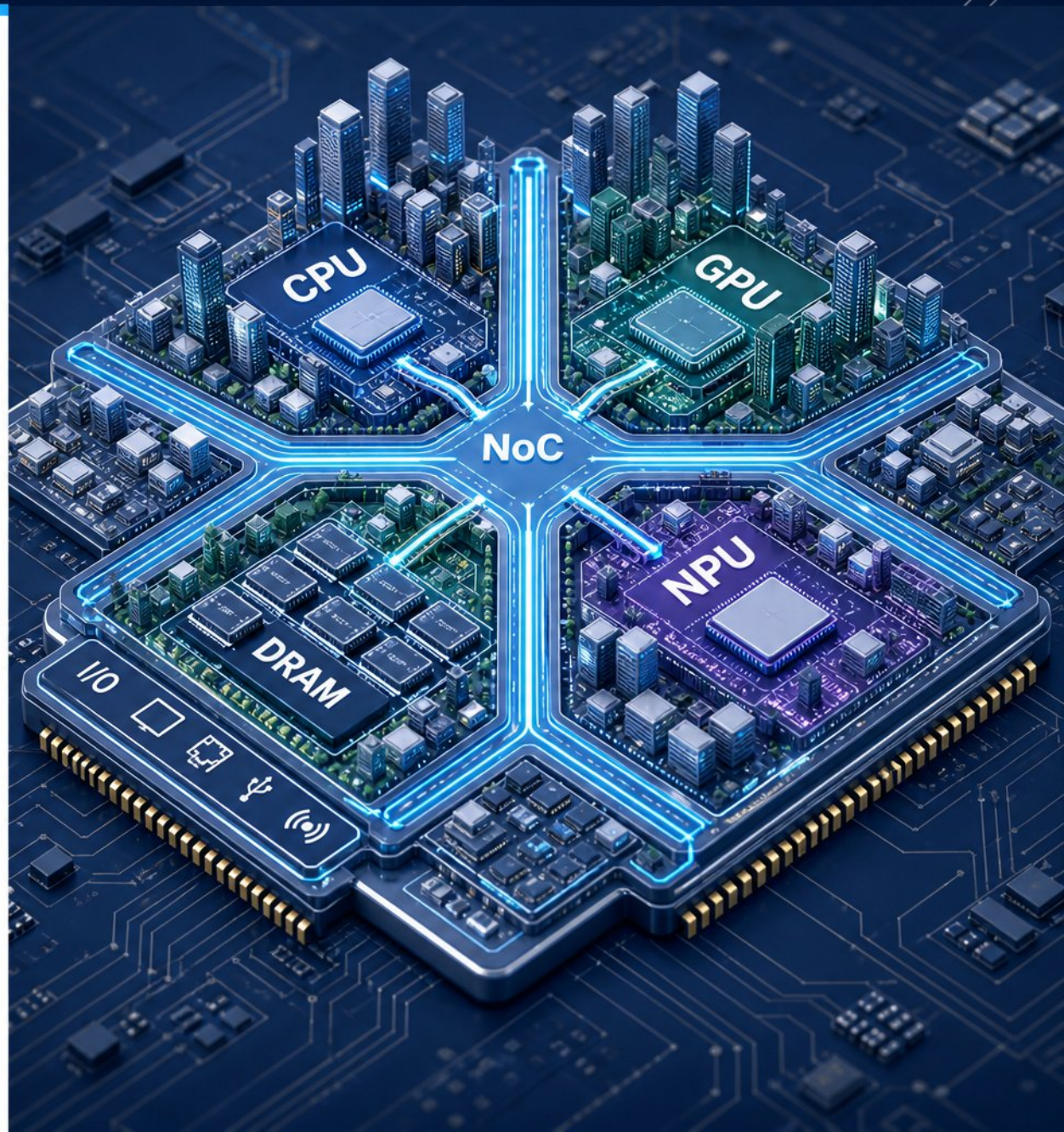
Communicating with the Outside World: I/O, Buses, and DMA

- - Memory-Mapped I/O (MMIO) turns device registers into memory addresses
- - On-chip buses (AMBA/AXI) and external links (PCIe, USB) move data
- - DMA engines copy data between memory and peripherals, bypassing the CPU
- - The IOMMU gives devices virtual memory, restricting unauthorized access



The Modern Landscape: System-on-Chip (SoC) Architecture

- SoC integrates CPU, GPU, memory, and I/O onto a single chip instead of separate boards
- On-chip interconnect (NoC) acts as highways between compute, memory, and I/O blocks
- Unified Memory Architecture (UMA) lets CPU and GPU share physical memory, eliminating copies
- Shared DRAM bandwidth creates bottlenecks; coherency rules govern multi-engine access
- Hardware-aware software design is critical: know your data paths and synchronization points



PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/a50420e5/public