A Technical Study of a Basic CPU Architecture

From Fundamental Blocks to the Instruction Cycle

Francisco Gonçalves
Co-authoring support: Augustus

February 08, 2026

Abstract

This paper presents a rigorous yet accessible description of a basic Central Processing Unit (CPU) architecture. It covers the control unit, arithmetic logic unit (ALU), registers, buses, clocking, memory interface, and status flags. The fetch-decode-execute cycle is detailed as the operational core of instruction processing. The paper also introduces interrupt handling, instruction set fundamentals, and baseline performance metrics. Although modern processors include advanced techniques such as pipelining, superscalar issue, and out-of-order execution, this study demonstrates that their conceptual foundation remains rooted in the same canonical model: control, datapath, memory, and synchronization.

Keywords: CPU architecture, control unit, ALU, registers, instruction cycle, von Neumann model, performance metrics

1. Introduction

A CPU is the computational core responsible for interpreting and executing program instructions. Despite the complexity of contemporary microarchitectures, the didactic model of a basic CPU remains the most effective entry point for architectural analysis and low-level optimization. In this model, execution proceeds through a recurring sequence: instruction fetch, decode, and execute.

2. Architectural Overview

A basic CPU can be represented by the following functional blocks:

- Program Counter (PC): holds the address of the next instruction.
- Instruction Register (IR): stores the currently fetched instruction.
- Control Unit (CU): decodes instructions and emits control signals.
- Arithmetic Logic Unit (ALU): executes arithmetic and logical operations.
- General-purpose registers: fast storage for operands and results.
- Status/Flags register: records condition codes (Zero, Carry, Sign, Overflow).
- Memory interface: coordinates data and instruction transfers.
- Clock: synchronizes all state transitions.

3. Core Functional Blocks

3.1 Control Unit

The Control Unit orchestrates processor behavior. It interprets operation codes, determines operand sources and destinations, and issues time-aligned control signals to registers, ALU, and memory interface. Two common implementation styles are hardwired control and microprogrammed control.

3.2 Arithmetic Logic Unit

The ALU is the computational engine of a basic CPU. Typical operations include ADD, SUB, AND, OR, XOR, comparisons, and shifts. Each operation may update status flags used by conditional branch instructions.

3.3 Register Set

Registers provide the fastest storage tier in the machine. Essential registers include PC, IR, SP (stack pointer), general-purpose registers, and a status register. Register-rich designs reduce memory traffic and improve effective throughput.

3.4 Buses and Internal Data Movement

Conceptually, a basic system uses data, address, and control buses. Bus width determines transfer granularity and maximum addressable memory space.

4. The Fetch-Decode-Execute Cycle

4.1 Fetch

The CPU places the PC value on the address bus, reads instruction memory, loads IR, and updates PC.

4.2 Decode

The Control Unit decodes opcode and addressing mode, selects datapath routes, and schedules micro-operations.

4.3 Execute

The selected execution unit performs the required operation. Results are written back to registers or memory, flags are updated, and control flow may alter PC.

5. Datapath and Micro-operations

The datapath includes register file ports, multiplexers, ALU inputs, temporary latches, and write-back paths. Instruction execution is decomposed into micro-operations distributed over clock cycles.

Example micro-sequence for ADD R3, R1, R2:

1. Read R1 and R2 from the register file.
2. Route both operands to ALU input ports.
3. Perform addition and update flags.

4. Write result into R3.

6. Memory and I/O Interface

In a minimal model, instructions and data share the same memory space (von Neumann organization). The memory interface controls read/write timing and handshaking. In practical systems, cache hierarchy reduces effective memory latency.

7. Interrupts and Exceptions

Interrupt support allows asynchronous events to be serviced without continuous polling. A typical sequence is: complete current instruction, save minimal context, branch to service routine, execute handler, and return.

8. Instruction Set Baseline

A basic ISA typically includes:

- Data movement: LOAD, STORE, MOV
- Arithmetic/logical: ADD, SUB, AND, OR, XOR, CMP
- Control flow: JMP, JZ, JNZ, CALL, RET
- System control: NOP, HLT, INT

9. Performance Metrics

Baseline evaluation of a basic CPU should track clock frequency, CPI (cycles per instruction), IPC (instructions per cycle), and memory latency. A classic execution-time relation is:

$$CPU\ Time = Instruction\ Count \times CPI \times Clock\ Period$$

10. From Basic to Modern CPU Designs

Advanced processors extend the basic model with pipelining, branch prediction, out-of-order execution, superscalar dispatch, SIMD/vector units, and multicore parallelism. Nevertheless, these enhancements preserve the same conceptual nucleus: control, datapath, memory, and synchronization.

11. Conclusion

A basic CPU architecture provides a rigorous conceptual framework for understanding real processors. Mastering this structure is essential for systems programming, compiler optimization, embedded design, and hardware-software co-design.

References

5. J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 6th ed., Morgan Kaufmann, 2019.
6. D. A. Patterson and J. L. Hennessy, Computer Organization and Design RISC-V Edition, 2nd ed., Morgan Kaufmann, 2020.
7. A. S. Tanenbaum and T. Austin, Structured Computer Organization, 6th ed., Pearson, 2012.
8. M. M. Mano and C. R. Kime, Logic and Computer Design Fundamentals, 5th ed., Pearson, 2015.
9. J. P. Hayes, Computer Architecture and Organization, 3rd ed., McGraw-Hill, 1998.