

System-Level Learning Roadmap (C++ → Compiler → OS → Hardware)

This roadmap helps you understand how your C++ program is compiled, stored, loaded into memory, and executed by the CPU. It connects Compiler Design, Operating Systems, Computer Architecture, and Systems Programming into one flow.

1■■ Major Subjects Involved

- 1 **Computer Organization & Architecture (COA):** How CPU, registers, memory, and buses work — helps you understand hardware-level execution.
- 2 **Operating System (OS):** How programs are loaded, executed, and managed — covers process creation, memory, and paging.
- 3 **Compiler Design:** How source code becomes machine code — preprocessing, parsing, and code generation.
- 4 **Systems Programming:** How OS services and low-level programs interact — linkers, loaders, and process execution.

■ Best Reference Books

- 1 **Computer Organization & Architecture:**
 - Computer Organization and Design – David A. Patterson & John L. Hennessy
 - Computer Organization – Carl Hamacher
- 2 **Operating Systems:**
 - Operating System Concepts – Abraham Silberschatz
 - Operating Systems: Three Easy Pieces (Free Online)
- 3 **Compiler Design:**
 - Compilers: Principles, Techniques, and Tools – Aho (Dragon Book)
 - Compiler Design in C – Allen Holub
- 4 **Systems Programming:**
 - Systems Programming and Operating Systems – Dhamdhere
 - System Software – Leland Beck
- 5 **All-in-One (Highly Recommended):**
 - Computer Systems: A Programmer's Perspective (CS:APP) – Randal Bryant & David O'Hallaron

■ Study Order (Zero → Expert)

1. C++ Programming → C++ Primer / E. Balagurusamy
2. Computer Architecture → Patterson & Hennessy
3. Operating Systems → Silberschatz / OSTEP
4. Compiler Design → Dragon Book / Holub
5. Systems Programming → Dhamdhere / Beck
6. Integration → CS:APP (Bryant & O'Hallaron)

■ Pro Tip

If your goal is to truly understand the flow from code to CPU, follow this learning path:

C++ → Computer Architecture → Operating System → Compiler Design → Systems Programming → CS:APP