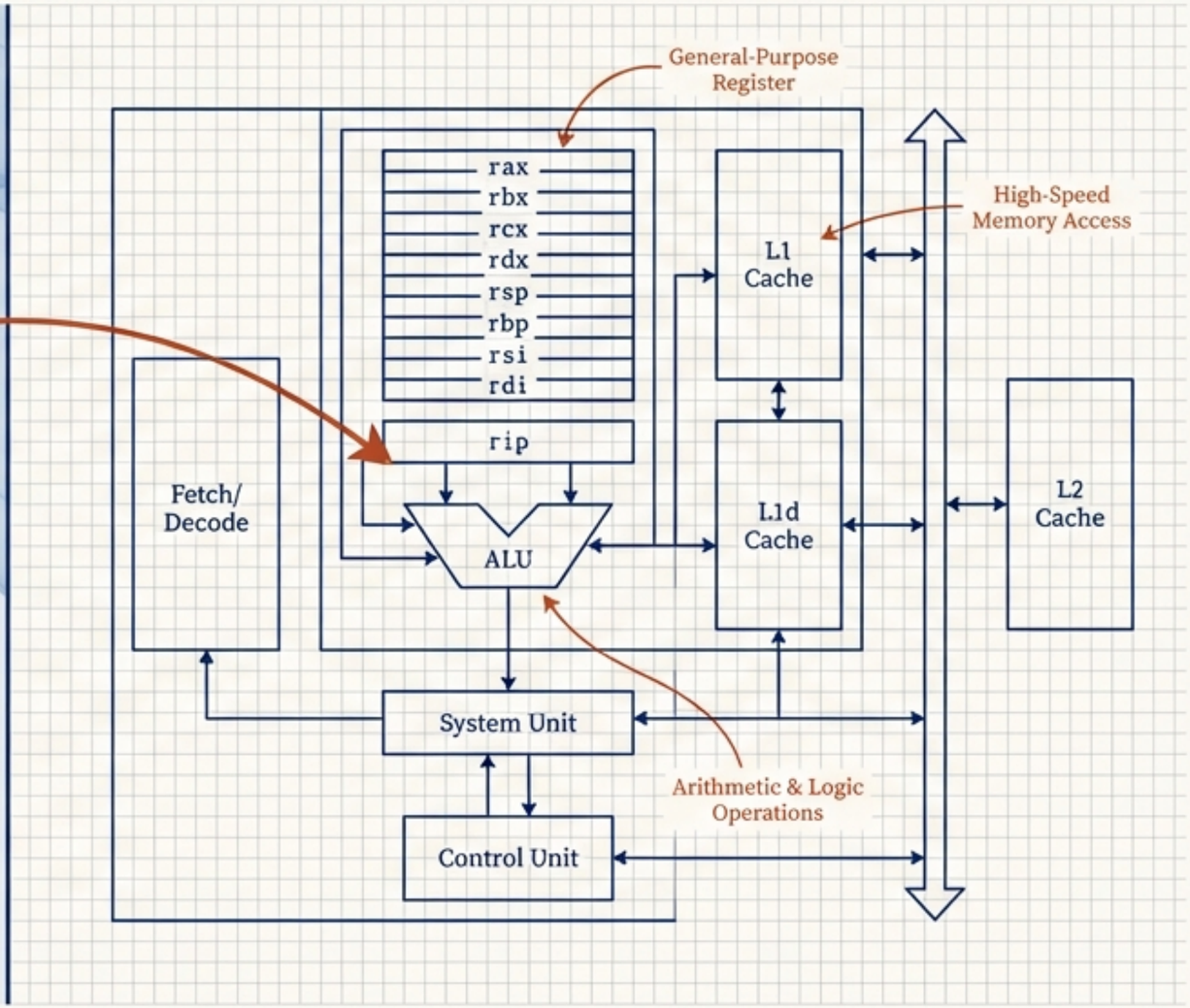
From High-Level Magic to Low-Level Mastery

An Introduction to CS 5008: Systems and Compilers

```
data = process(source)
```



You’re a Programmer. You Make Things Happen.

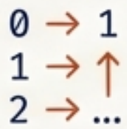
You know how to program in languages like Python and Java. You understand variables, loops, functions, and objects. You are used to powerful abstractions that work like magic.

```
my_list = [1, 2, 3]

my_list.append(4)

print(f"Size: {len(my_list)}")
```

What Python Does For You

Allocates memory automatically

Manages memory with a garbage collector

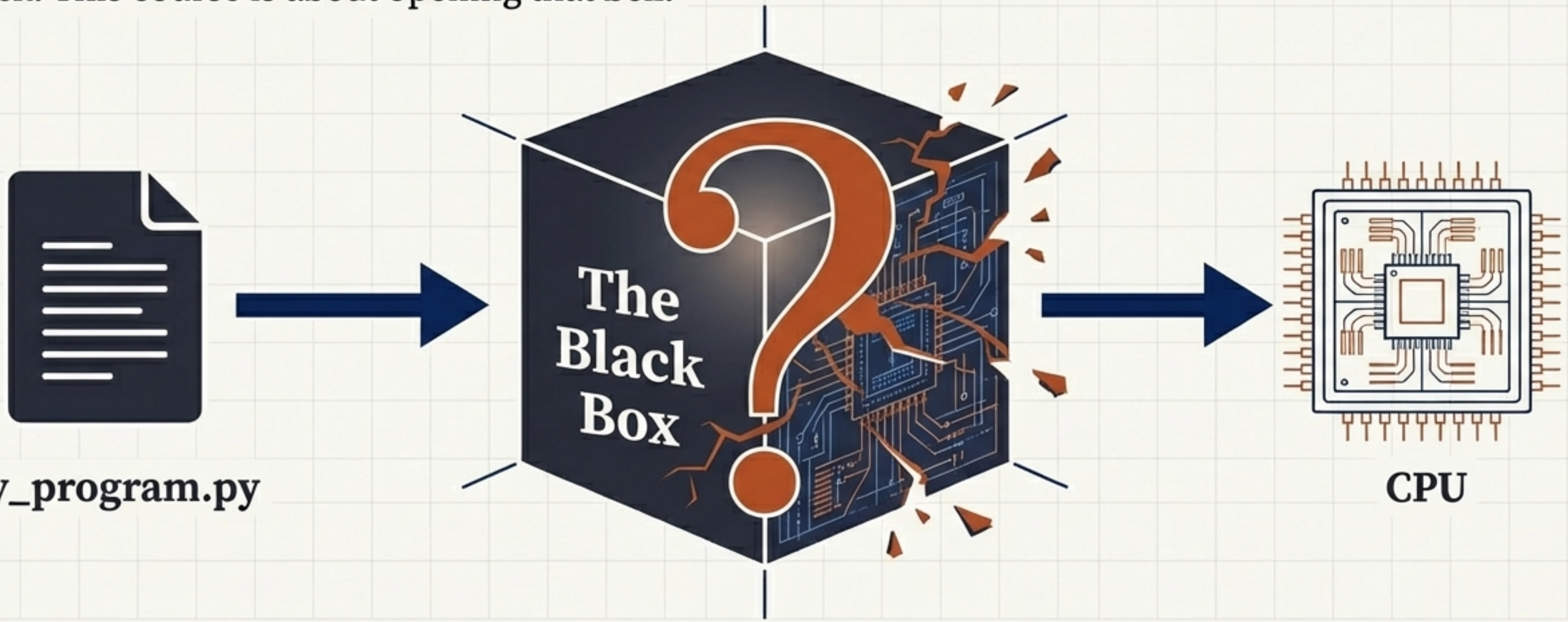


Tracks list size for you

Handles errors gracefully

But What Lies Beneath the Abstractions?

High-level languages are powerful, but they hide the fundamental workings of the machine. This limits your ability to write the highest performance code and understand the full technology stack. This course is about opening that box.



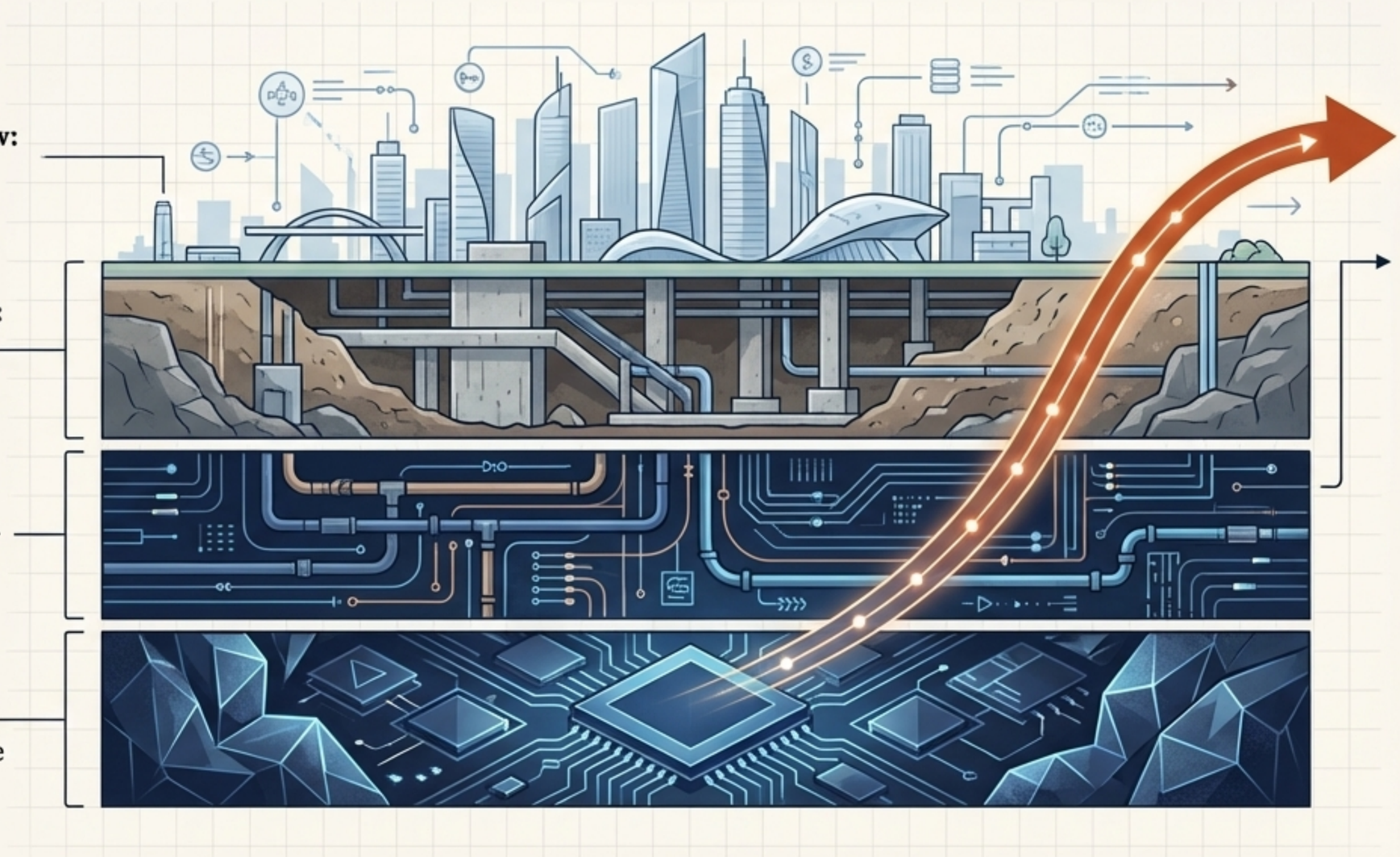
Our Journey: Down the Stack, Then Back Up With New Mastery

The World You Know:
Python / Java

The Descent Begins:
C Programming - The Lingua Franca of Systems

Deeper Control:
Memory & Pointers - Direct Hardware Manipulation

The Foundation:
x86-64 Assembly - The Native Language of the Machine



You Will Build a Complete Compiler

Step 1: Thinking in C

We learn C not in a vacuum, but by building on what you already know.

YOU KNOW: Python/Java
High-Level Abstractions

```
my_list = [1, 2, 3]
```

- Memory is managed automatically.

IN C
You Control Everything

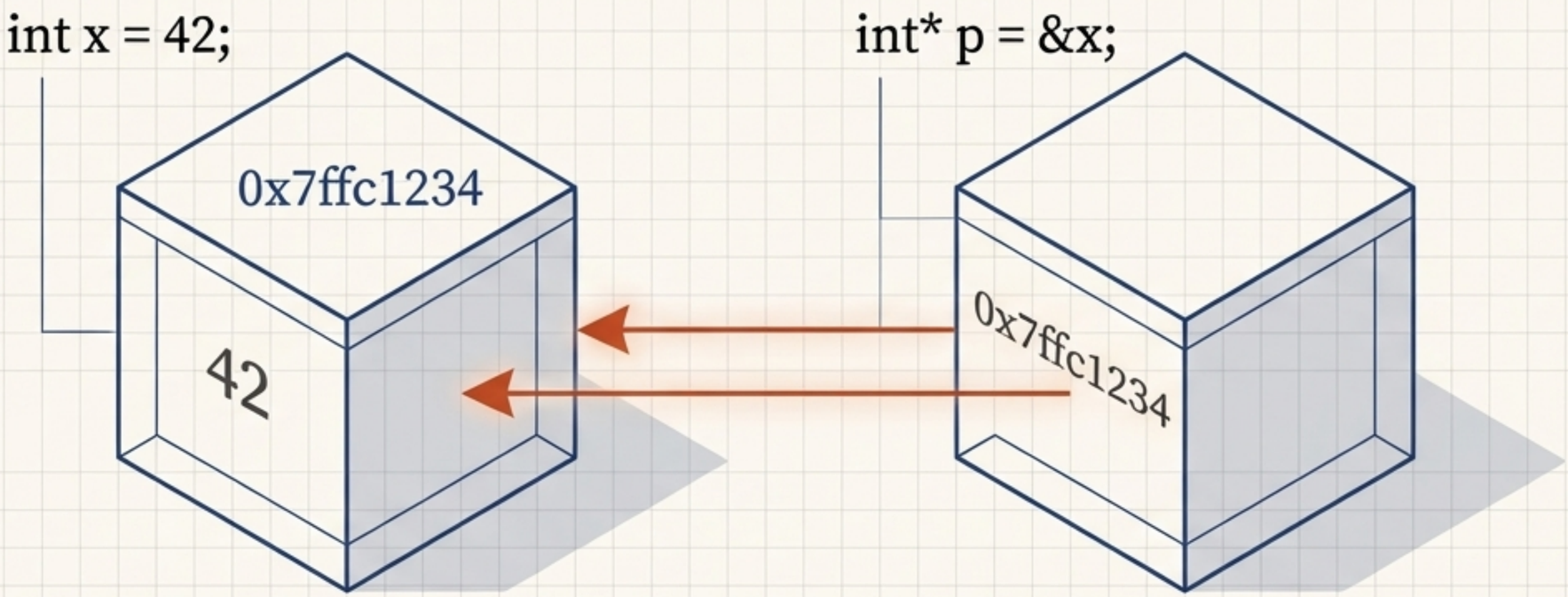
```
int* my_array = malloc(3 * sizeof(int));
int* my_array = {
    return 0;
}
```

- Memory is allocated and freed manually.

C gives you unparalleled performance and control by making you manage the details that Python and Java hide.

Pointers Aren’t Magic, They’re Addresses

Think of computer memory as a giant apartment building. Every byte has a unique address, like an apartment number. A variable lives at an address. A pointer simply stores that address.



& Get the apartment number.

***** Use the number to get what’s inside the apartment.

Reaching the Foundation: The Language of the CPU

Your compiler will translate high-level code into x86-64 assembly. Understanding it is key to debugging and seeing what your code *actually* does.

C Code

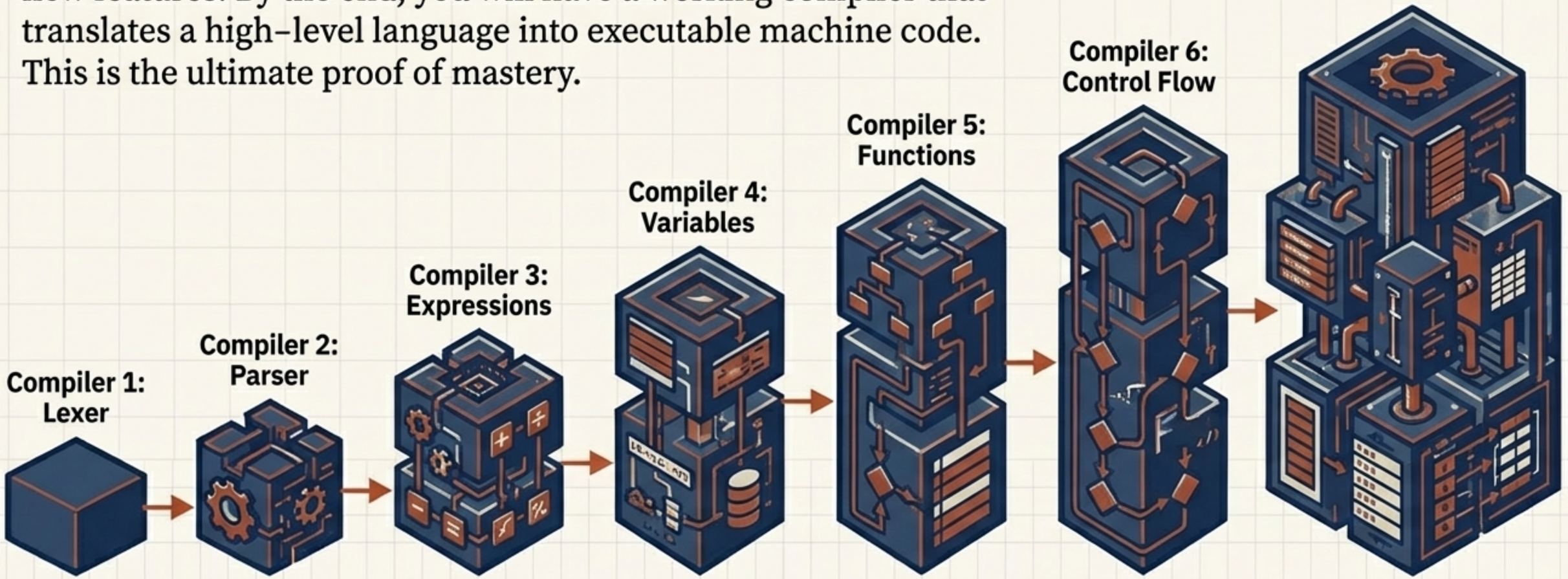
```
int get_answer()
{
  return 42;
}
```

x86-64 Assembly

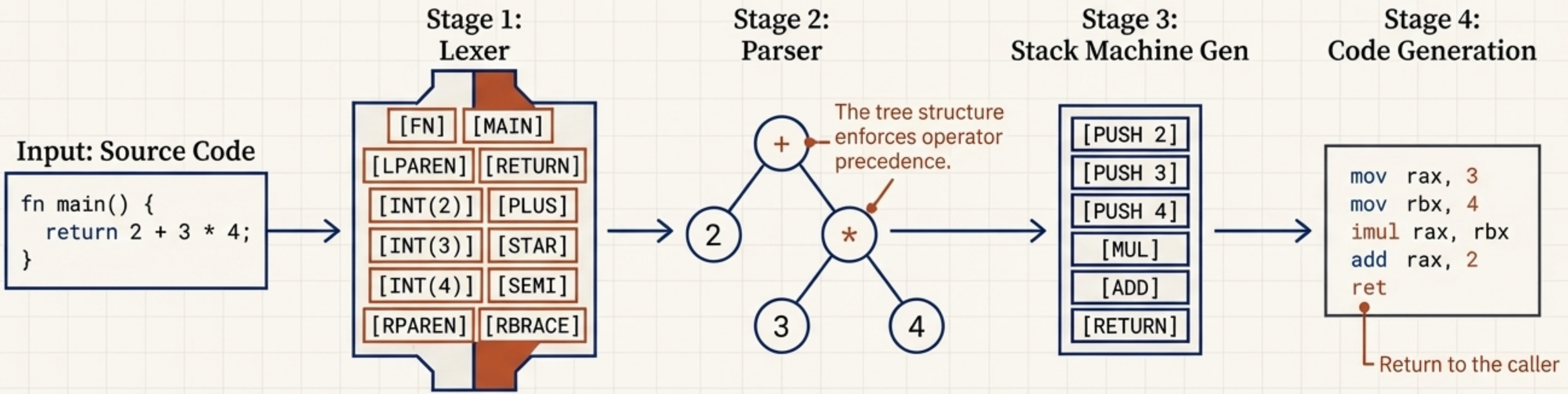
```
get_answer:
  mov rax, 42
; Annotation: "rax is the register for return values"
  ret
; Annotation: "Return to the caller"
```

The Ascent: You Will Build a Complete Compiler

Throughout the semester, you will build 7 compilers, each adding new features. By the end, you will have a working compiler that translates a high-level language into executable machine code. This is the ultimate proof of mastery.



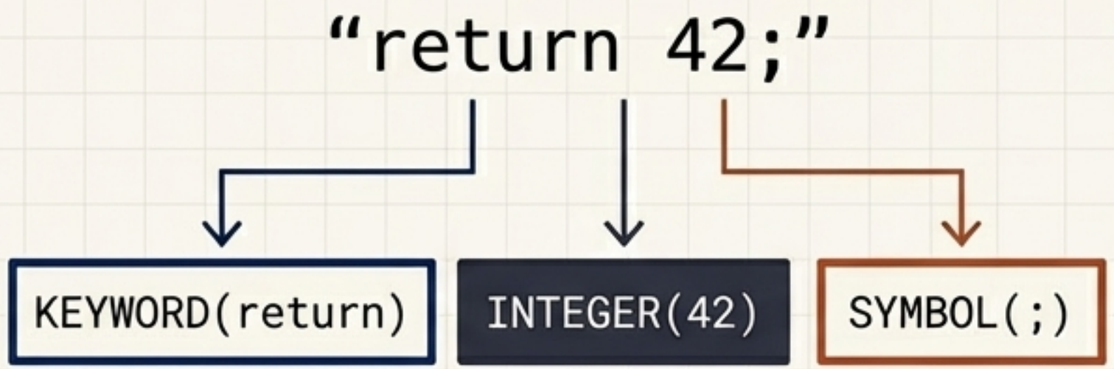
The Compiler Pipeline: From Source Code to Machine Code



Stage 1 & 2: Finding the Words, Then the Meaning

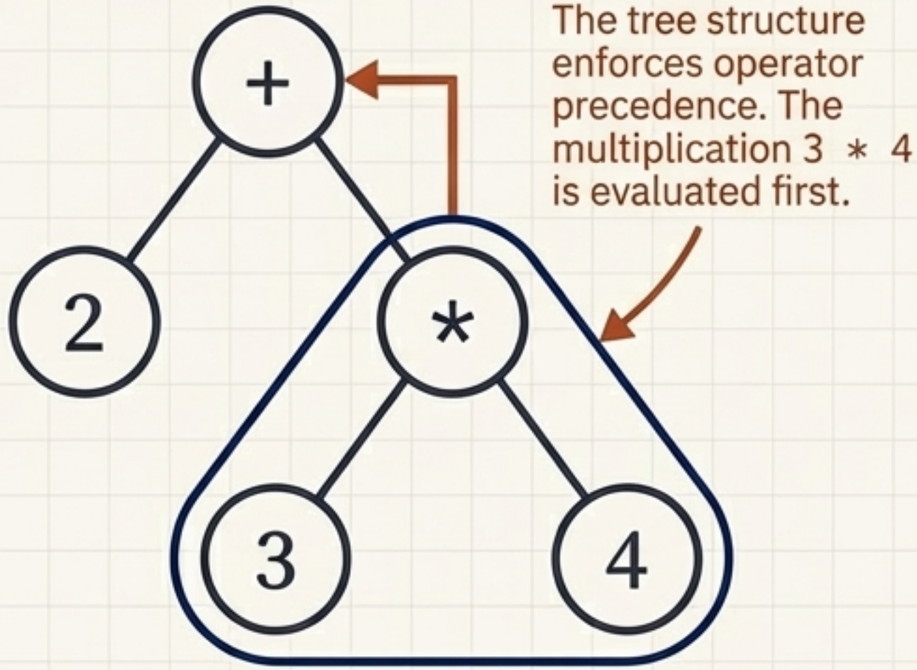
From Characters to Tokens

Like grouping letters into words.



From Tokens to Structure

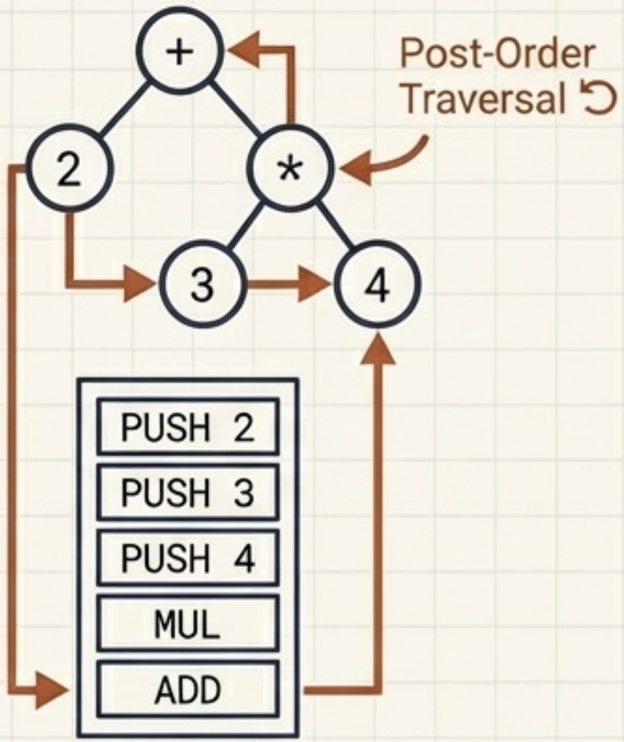
“Dog bites man” vs. “Man bites dog.” Same words, different structure, different meaning.



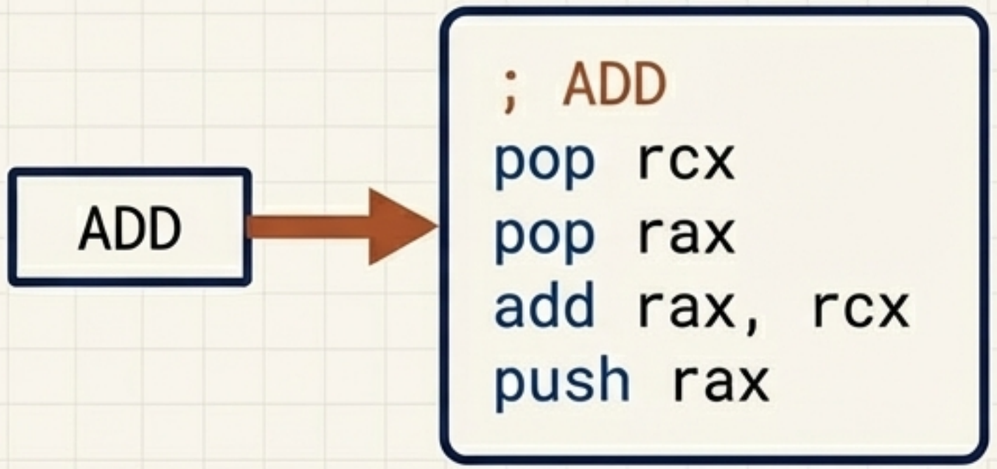
Stage 3 & 4: From a Simple Plan to a Real Blueprint

A Simple, Platform-Independent Plan

Like an RPN calculator (2 3 4 * +' means 2 + (3 * 4)').



Translating the Plan into Assembly



Each simple stack operation is mechanically translated into a sequence of concrete machine instructions.

Your 15-Week Build Plan

Phase 1: The Foundations (Weeks 1-4)

C Fundamentals, Memory & Pointers, x86-64 Assembly, Data Structures

Mastering the tools and the terrain.

Phase 2: The Core Build (Weeks 5-10)

Compiler Pipeline, Lexer, Parser, Expressions, Variables, Functions, Control Flow

Constructing the compiler, feature by feature.

Phase 3: Advanced Systems (Weeks 11-15)

Strings & Arrays, Dynamic Memory, Graphs, Advanced Topics, Exam Prep

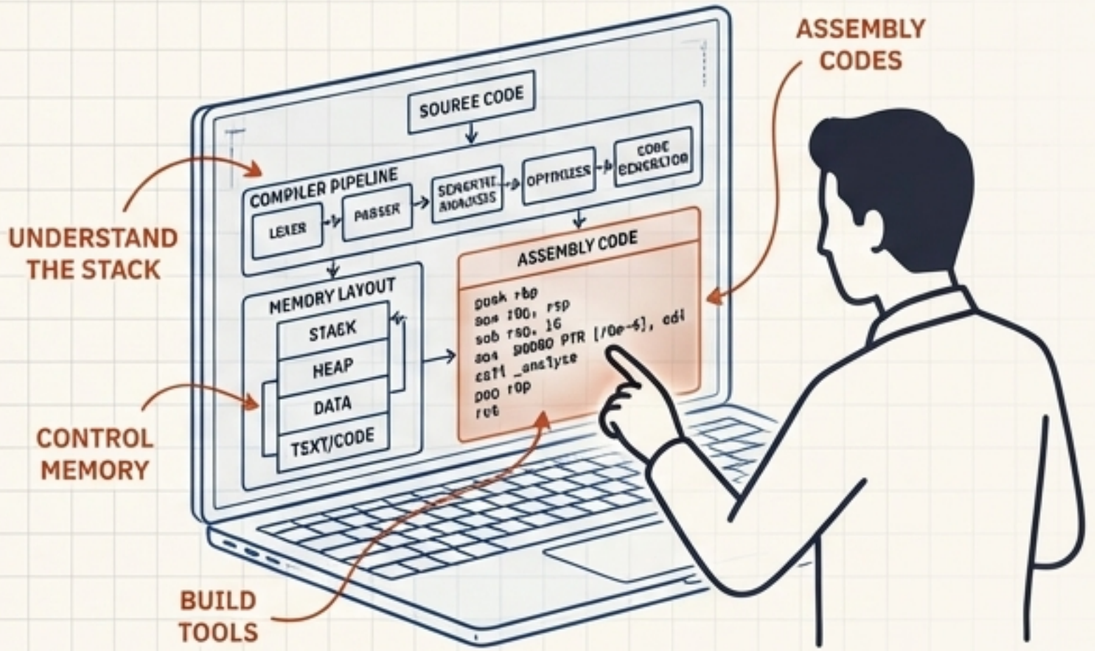
Adding power features and proving your mastery.

You Start as a User of Abstractions. You Emerge an Architect of Systems.

BEFORE CS 5008

- Writes high-level code.
- Relies on ‘magic’ black boxes.
- Is limited by the language’s design.

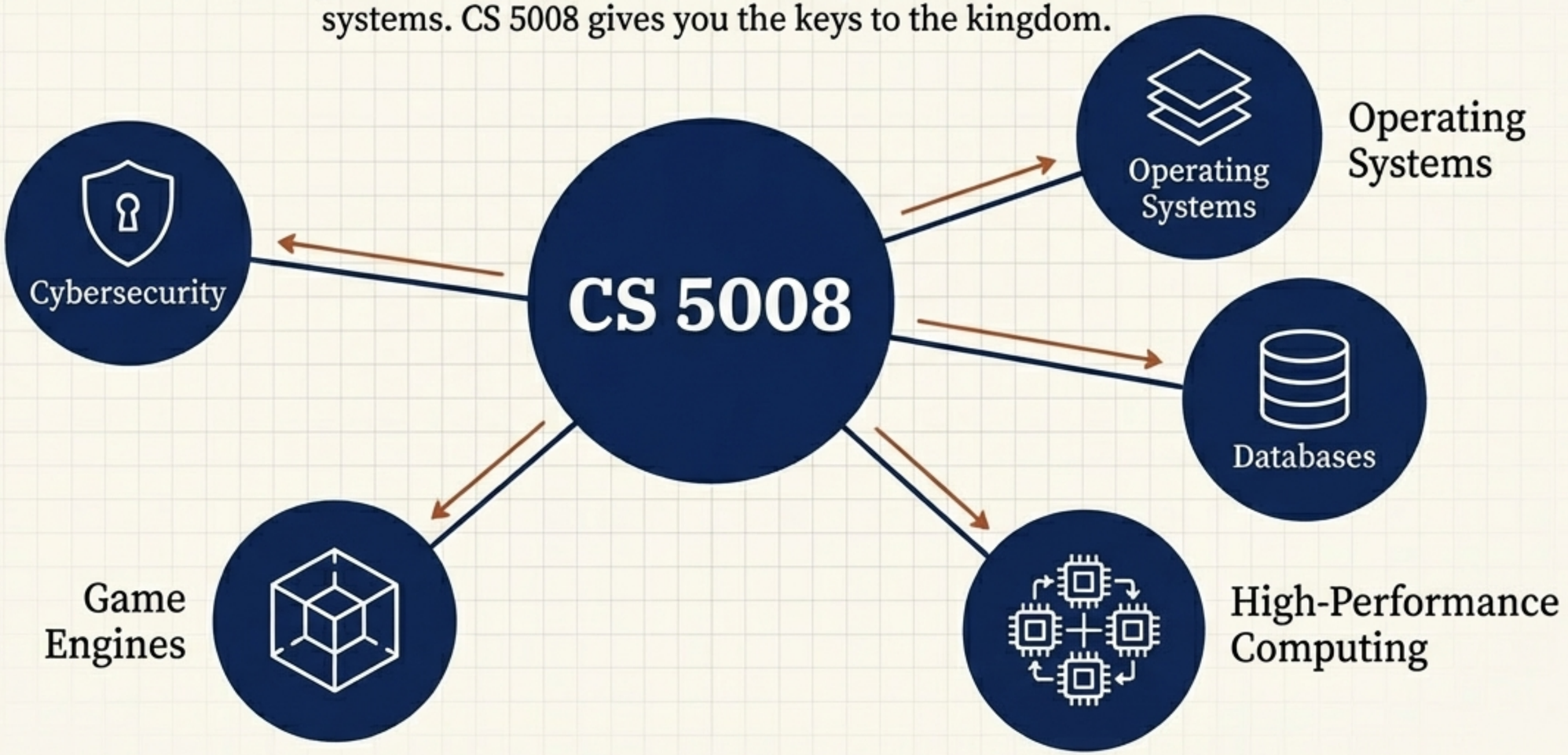
AFTER CS 5008



- Understands the full stack, from source to silicon.
- Controls memory and performance directly.
- Is capable of building foundational tools like compilers, OS, and databases.

This is the Foundation For Everything That Comes Next

Understanding systems and compilers isn't just an academic exercise. It's the key to unlocking high-performance computing, cybersecurity, database design, game engine development, and operating systems. CS 5008 gives you the keys to the kingdom.



Your Journey Begins

Let’s build.