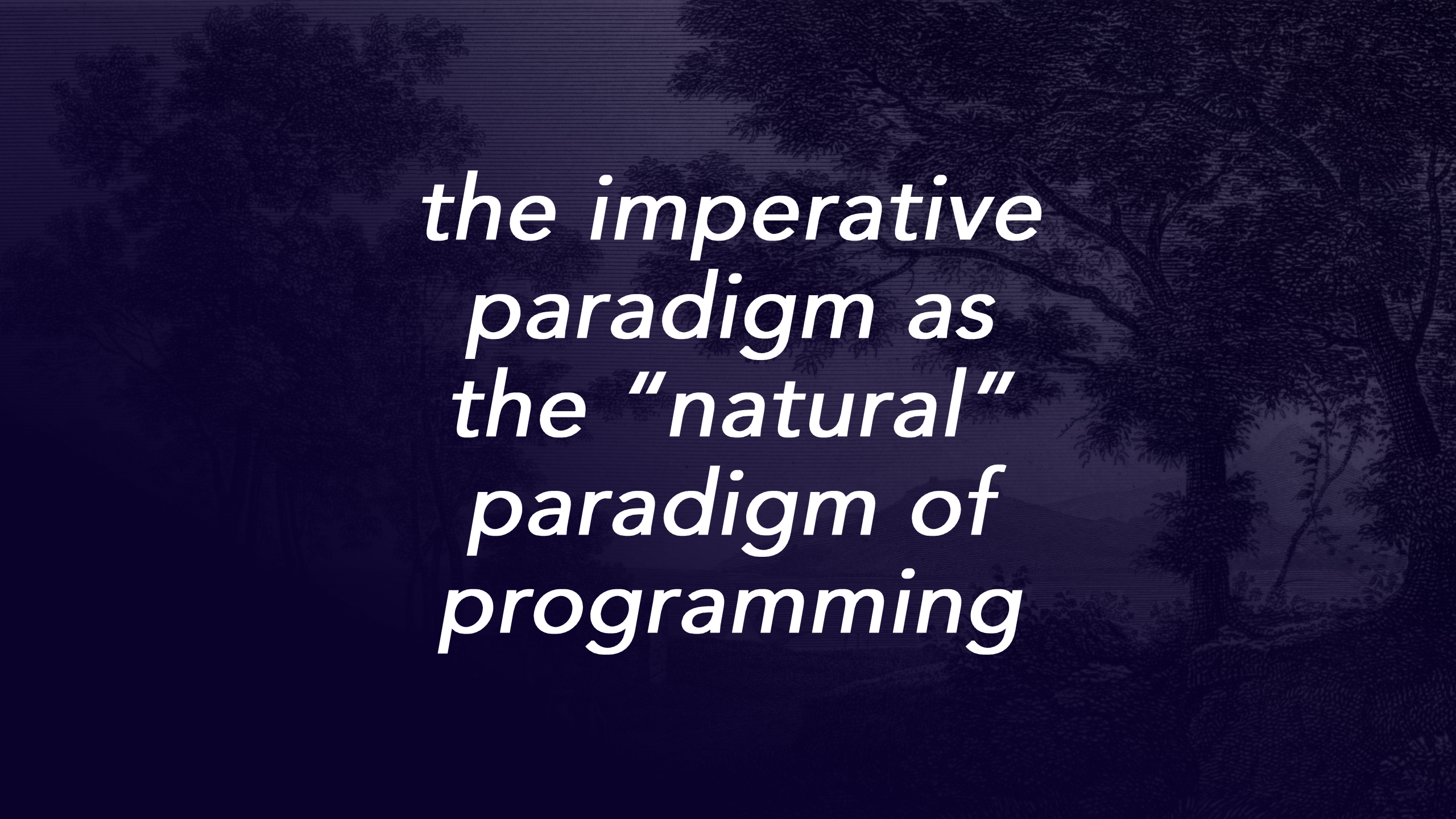

Imperative
Programming
Paradigm


the imperative
paradigm as
the “natural”
paradigm of
programming

Machine Code in Assembly

```
INC ITER
MOV AH, 7
ADD AH, AL
```


The STATE
The existence of an explicit state is the foundation of imperative programming.

The STATE

The **state** of a program or a process on a given instance is the snapshot of its immediate relevant environment and context.

Examples of state in different levels of programming

- **CPU state** - registers and relevant memory
- **Process** - the values inside the memory addresses it resides in
- **Program** - the conceptual set of variable values related to the program's runtime

Program state

```
int x = 3
int y = 4
x = x + y
```

Program state

```
int x = 3
int y = 4
x = x + y
```

VARIABLE	VALUE
x	3

Program state

```
int x = 3
int y = 4
x = x + y
```

VARIABLE	VALUE
x	3
y	4

Program state

```
int x = 3
int y = 4
x = x + y
```

VARIABLE	VALUE
x	7
y	4

Assignment Statements

One of the most important constructs of the imperative programming paradigm is the assignment statement.

Assignment Statements

Assignment statements lets you mutate the values of your variables.

Every assignment statement will correspond to new state for the program.

Assignment Statements

Assignment statements are usually executed using the "=" operator (some languages like Pascal use ":=" instead).

Although it borrows the equality operator from math, assignment operators behave very differently from an equality statement.

Assignment Statements

Instead of communicating some kind proposition, the assignment statement has an **imperative mood**.

Assignment Statements

An equality $a = b$ in math declares that some a is b, while an assignment operator a = b commands that a's value is now the same as b.

Mutation is introduced once you perform an assignment to a again, signifying a ***change*** in the value of a.

Assignment Statements

Imperative programming is characterized by imperative statements. Statements that tell the computer what to do.

Assignment Statements

The most important type of these statements is the assignment statement.

An assignment statement's effect to your computer is characterized by the progression from one state to another.

Assignment Statements

Assignment statements make states, and if you combine many of these assignment statements arranged in a particular manner, you can create a meaningful program that does something for you.

Structured Program Theorem

One of the theoretical frameworks proposed to characterize imperative programming is **Böhm-Jacopini theorem** also known as structured program theorem.

Structured Program Theorem

The theorem describes a formalism of a class called control flow graphs (flow charts) which are capable of representing any computable function.

Three ways of creating structured programs

1. Executing one subprogram, and then another subprogram **(sequence)**
2. Executing one of two subprograms according to the value of a boolean **(selection)**
3. Repeatedly executing a subprogram as long as a boolean expression is true **(iteration)**

Structure Programming

Structured programming enjoyed a universal popularity in computer science.

The constructs described by this formalism became the natural architecture for programming language designers.

Subparadigms under the imperative family

- Procedural Programming
- Object Oriented Programming

Procedural Programming

Programming languages like Fortran, ALGOL, BASIC, and C fall under the procedural paradigm. Languages under this paradigm simplify a complex system by subdividing a program into different **procedures** or functions.

Object Oriented Programming

Object oriented programming focuses on modelling a system based on the real-world ontology of objects. It uses an expressive type system to program the interactions within a system. Programming languages like C++, C#, Java, and Python are under this family.