Introduction

- From the course description:
  “This course covers the component technologies used in implementing modern programming languages, shows their integration into a system, and discusses connections between the structure of programming languages and their implementations.”

- Questions we will explore
  - How does a piece of program text end up getting executed on a machine?
  - What is the difference between a physical machine and a virtual machine?
  - How and when is a program transformed into a form amenable to machine execution?
  - How are various programming language features mapped to the capabilities of the machine?
  - What are the mathematical underpinnings of program translation?

Scope

- Focus
  - Automatic techniques for analyzing and transforming programs written in a given source language with the objective of executing them correctly on a target machine.
  - Sequential, imperative, class-based source language.

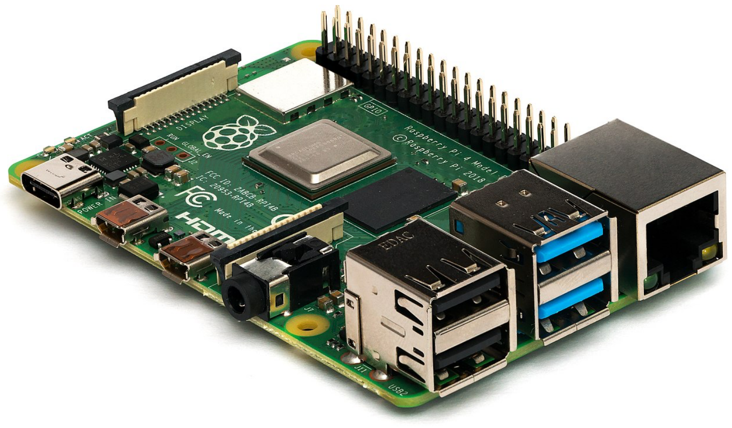
Scope

- Focus
  - Automatic techniques for analyzing and transforming programs written in a given source language with the objective of executing them correctly on a target machine.
  - Sequential, imperative, class-based source language.
- Out of scope
  - Code optimization.
  - Robust error handling.
  - Other language models, e.g., concurrent, functional, multi-paradigm, …
  - Other applications, e.g., debugging, verification, malware detection, …

Scope

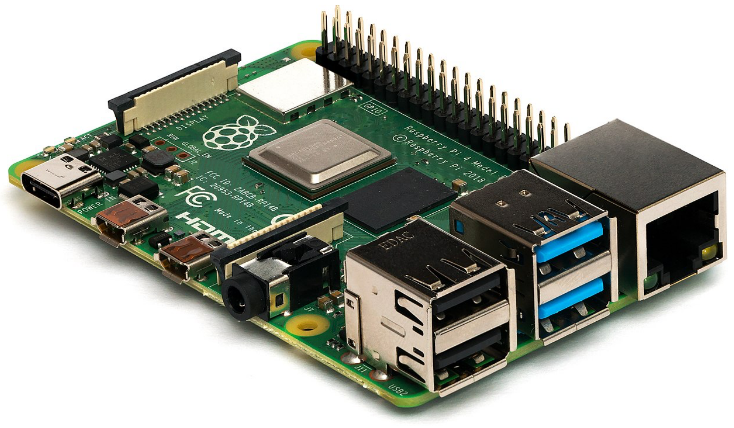
- Focus
  - Automatic techniques for analyzing and transforming programs written in a given source language with the objective of executing them correctly on a target machine.
  - Sequential, imperative, class-based source language.
- Out of scope
  - Code optimization.
  - Robust error handling.
  - Other language models, e.g., concurrent, functional, multi-paradigm, …
  - Other applications, e.g., debugging, verification, malware detection, …
- Analysis framework
  - Cost: Generally time, space, or energy.
  - Benefit: High performance, low memory use, energy efficiency, portability.
  - Risk: Correctness, security.

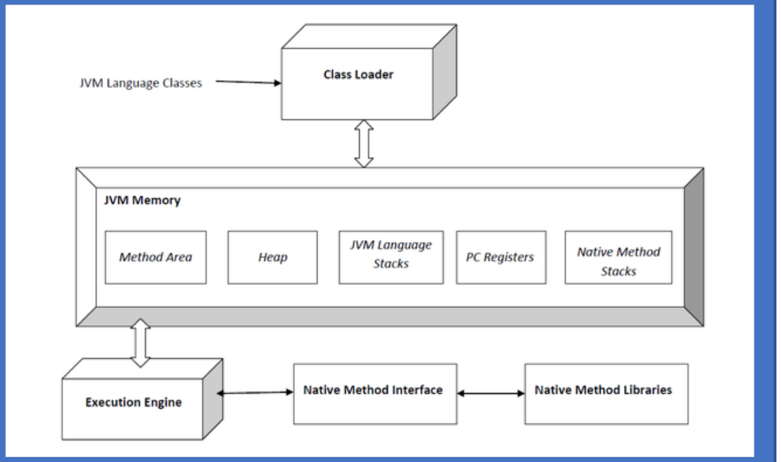
Execution Models



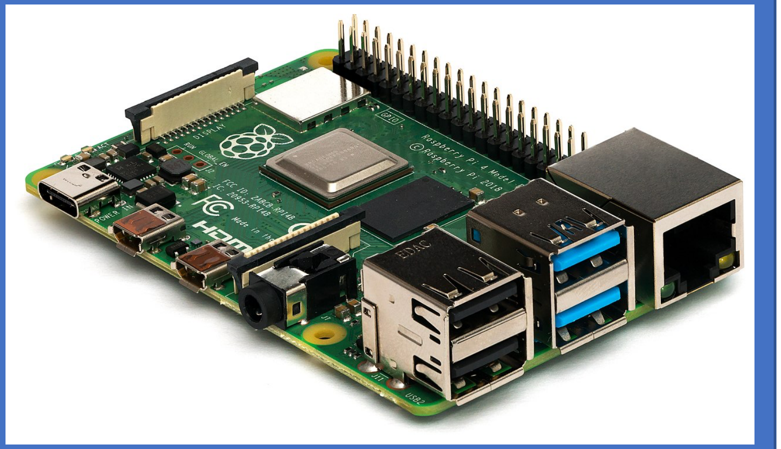
Physical Machine

Execution Models

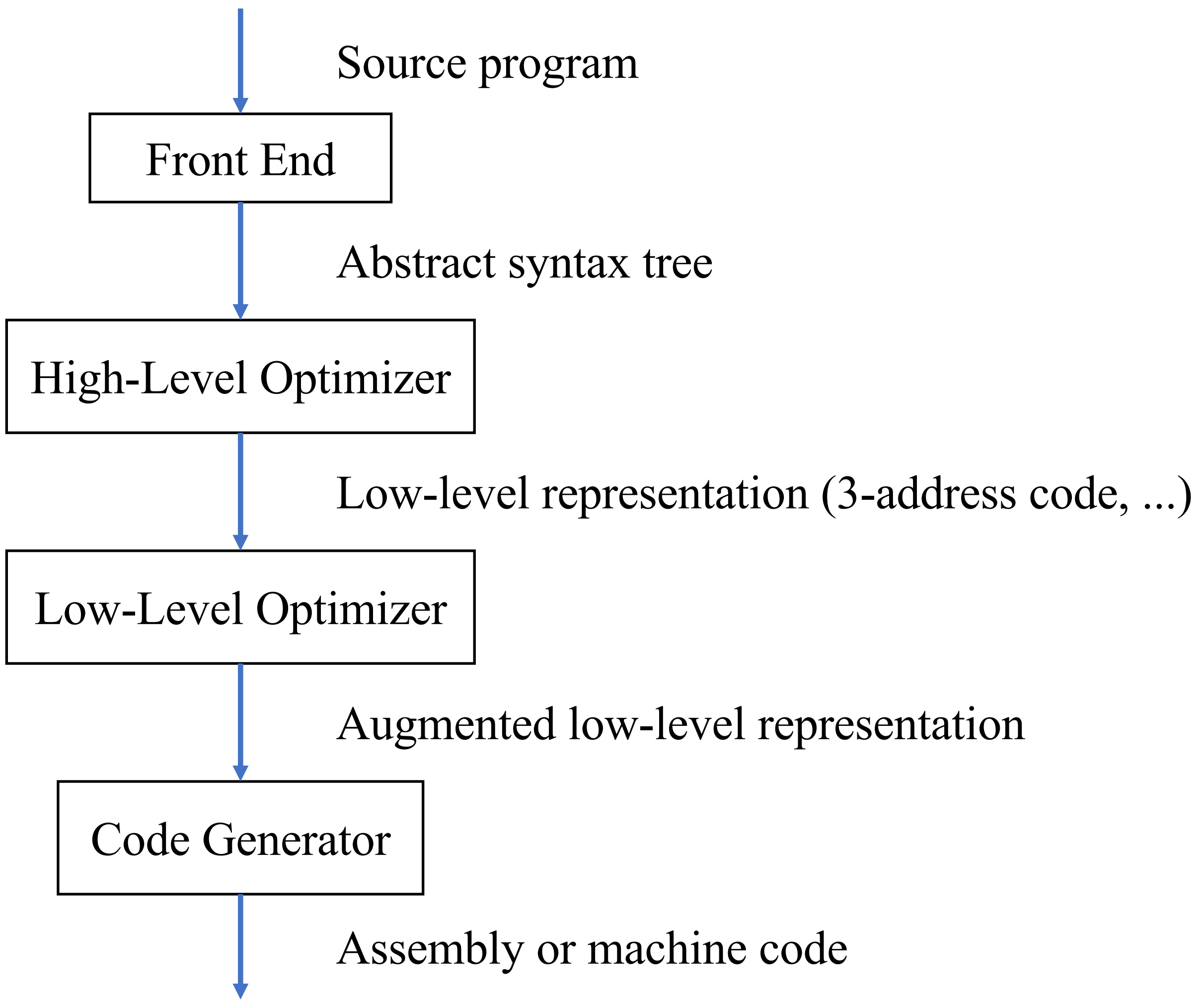

Physical Machine



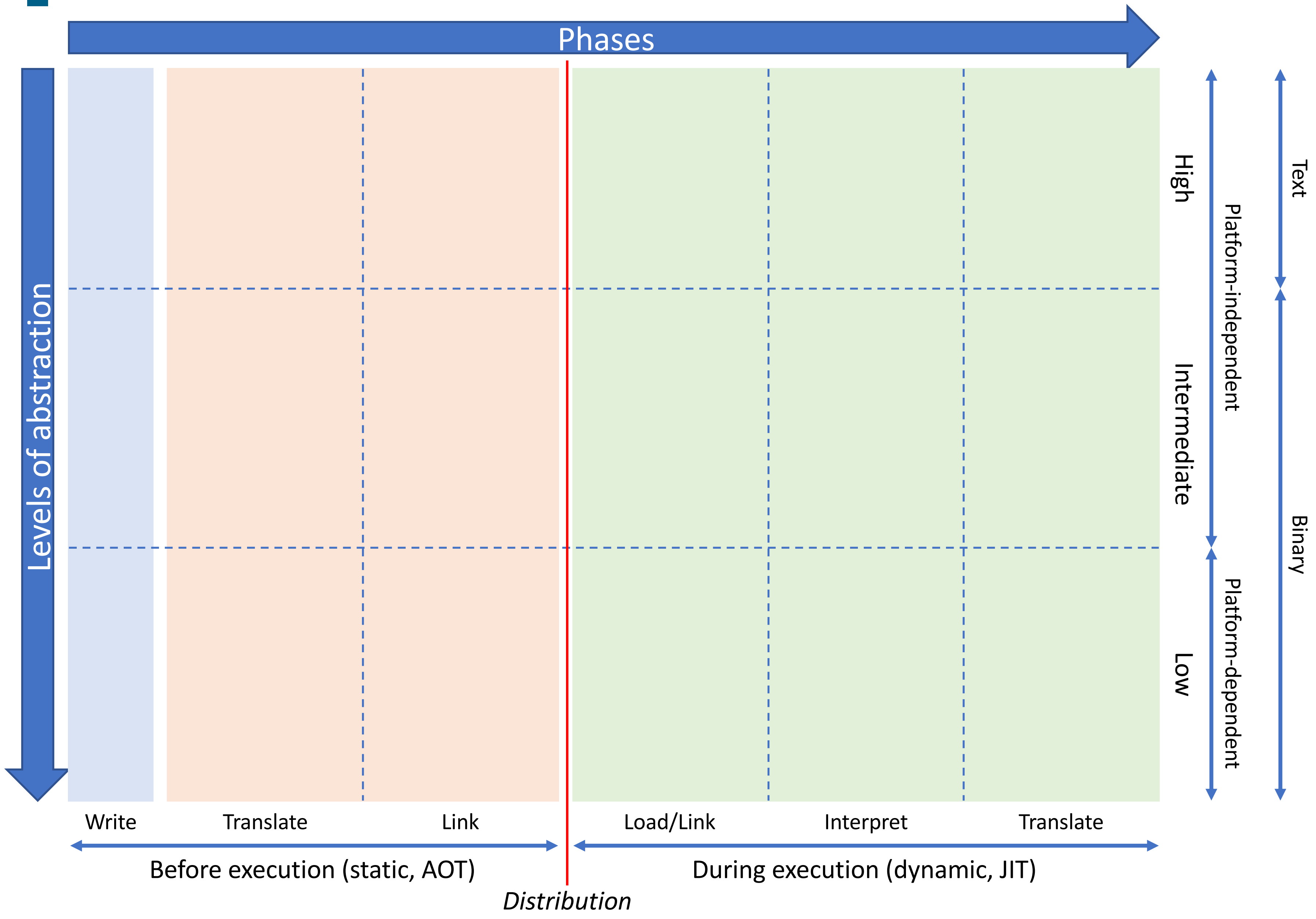
By Michelle Ridomi - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35963523


Virtual Machine

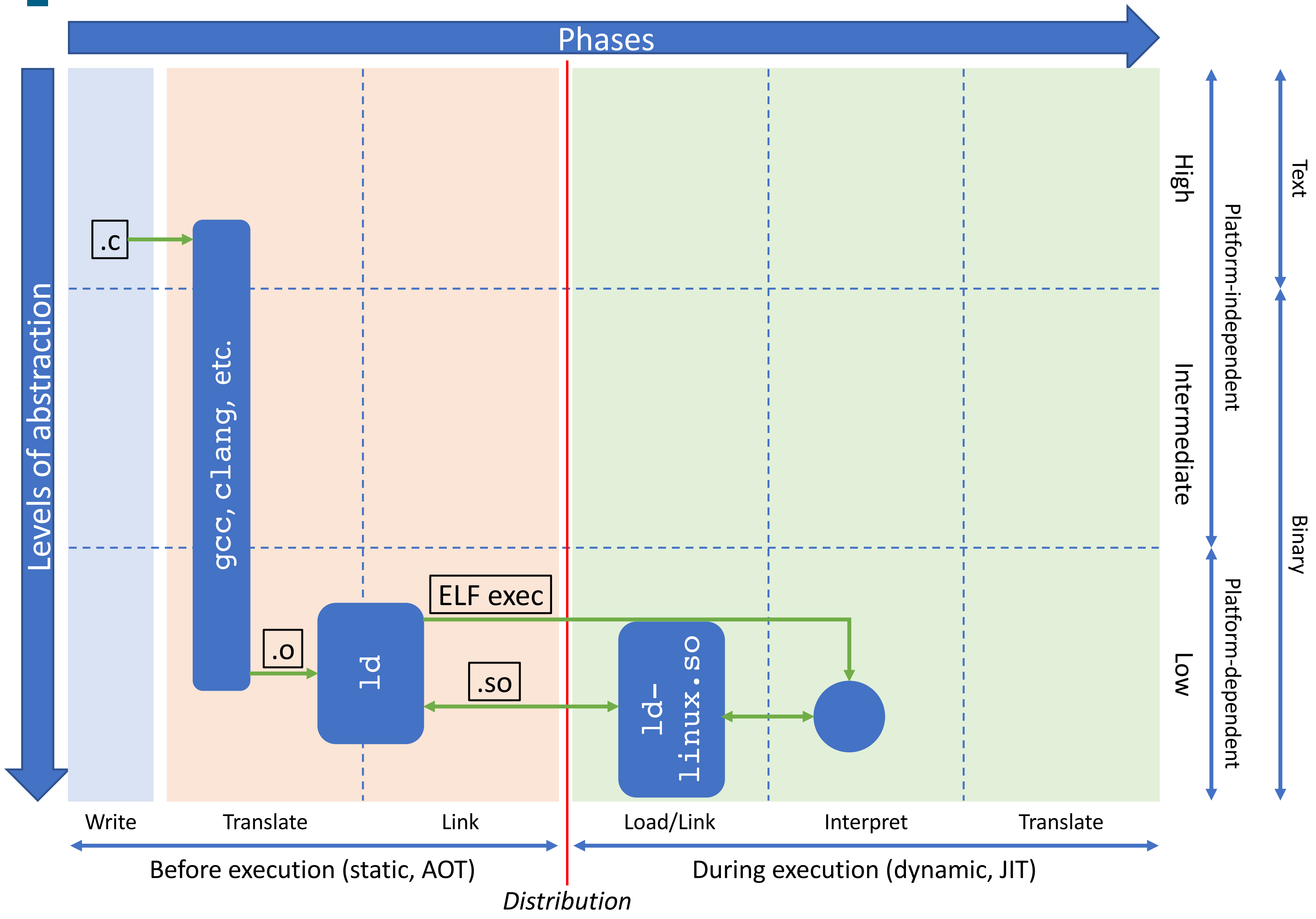
Transformation Techniques and Workflow



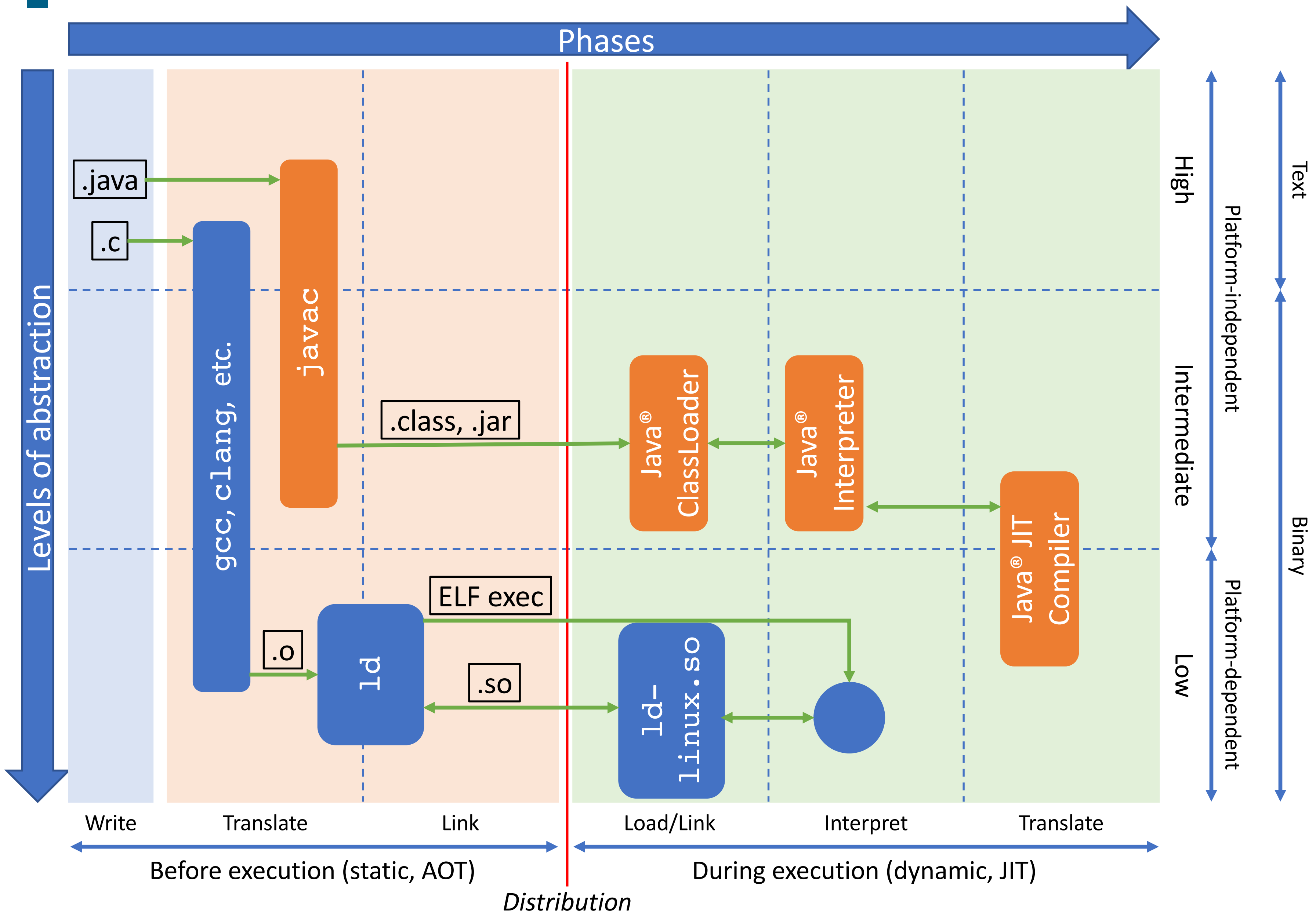
Binding Time: What Happens When



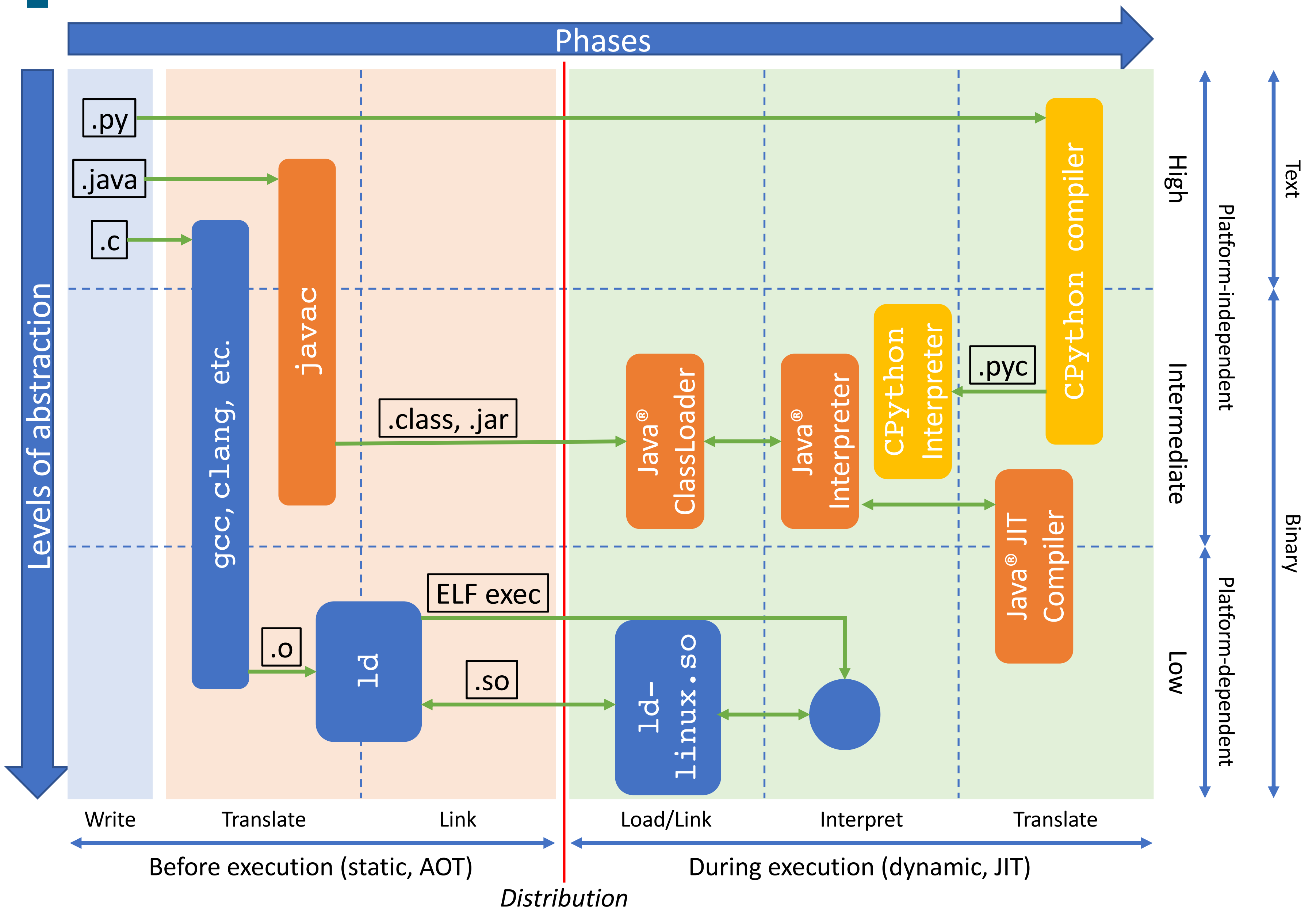
Binding Time: What Happens When



Binding Time: What Happens When



Binding Time: What Happens When



Languages, Grammars, Recognizers

- Language
    - (Webster's New Universal Unabridged Dictionary) language, n. *Any system of formalized symbols, signs, gestures, or the like, used or conceived as a means of communicating thought, emotion, etc.*
    - Mathematically: A language $\mathcal{L}$ is a set of strings over some alphabet Σ. Languages of interest are infinite, and therefore cannot be enumerated.

Languages, Grammars, Recognizers

- Language
    - (Webster's New Universal Unabridged Dictionary) language, n. *Any system of formalized symbols, signs, gestures, or the like, used or conceived as a means of communicating thought, emotion, etc.*
    - Mathematically: A language $\mathcal{L}$ is a set of strings over some alphabet Σ. Languages of interest are infinite, and therefore cannot be enumerated.
- Grammar
    - A grammar $\mathcal{G}(\mathcal{L})$ is a formal system that provides a finite generative description of $\mathcal{L}$.
    - $\mathcal{G}$ is a finite set of rules whose systematic repeated application will generate those, and only those, strings that belong to $\mathcal{L}$.

Languages, Grammars, Recognizers

- Language
  - (Webster's New Universal Unabridged Dictionary) language, n. *Any system of formalized symbols, signs, gestures, or the like, used or conceived as a means of communicating thought, emotion, etc.*
  - Mathematically: A language $\mathcal{L}$ is a set of strings over some alphabet Σ. Languages of interest are infinite, and therefore cannot be enumerated.
- Grammar
  - A grammar $\mathcal{G}(\mathcal{L})$ is a formal system that provides a finite generative description of $\mathcal{L}$.
  - $\mathcal{G}$ is a finite set of rules whose systematic repeated application will generate those, and only those, strings that belong to $\mathcal{L}$.
- Recognizer
  - A recognizer $\mathcal{R}(\mathcal{L}, w)$ is an automaton (i.e., machine) that can decide whether $w \in \mathcal{L}$.