AI编译器系列

基本介绍

ZOMI

About 关于本内容

1. **基本介绍**
   - Compiler and interpreter - 编译器与解释器
   - Just in time and Ahead of time - JIT和AOT编译方式
   - Pass and IR - Pass和中间表达IR
2. **传统编译器**
   - History of Compiler - 编译器的发展
   - GCC process and principle – GCC 编译过程和原理
   - LLVM/Clang process and principle – LLVM 编译过程和原理
3. **AI编译器**
   - History of AI Compiler – AI编译器的发展
   - Base Common architecture – AI编译器的通用架构
   - Different and challenge of the future – 与传统编译器的区别，未来的挑战与思考

Question

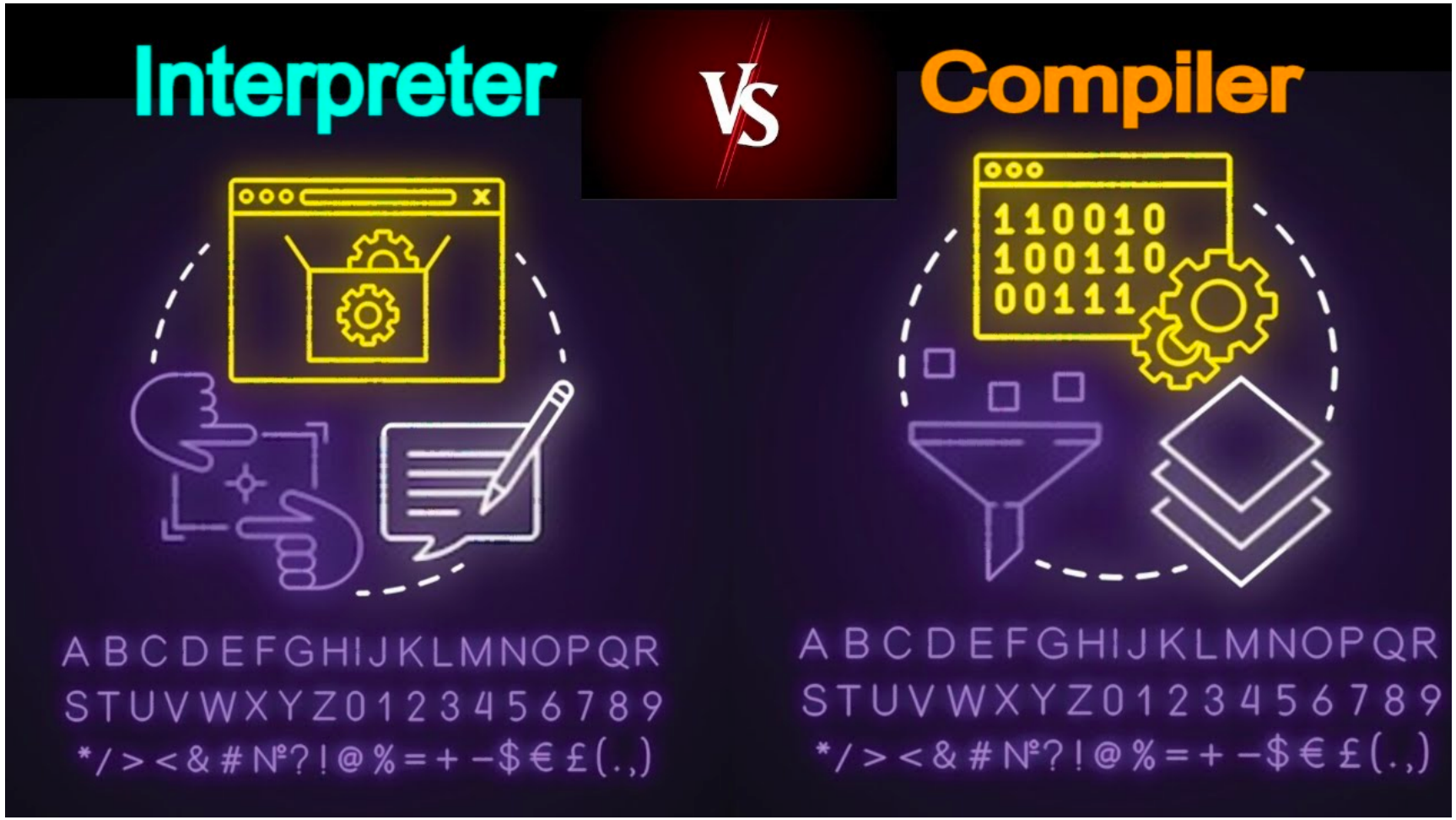
- Why need Compiler ?

  为什么需要编译器？

- Relationship between AI frameworks and AI compilers?

  AI框架和AI编译器之间什么关系？

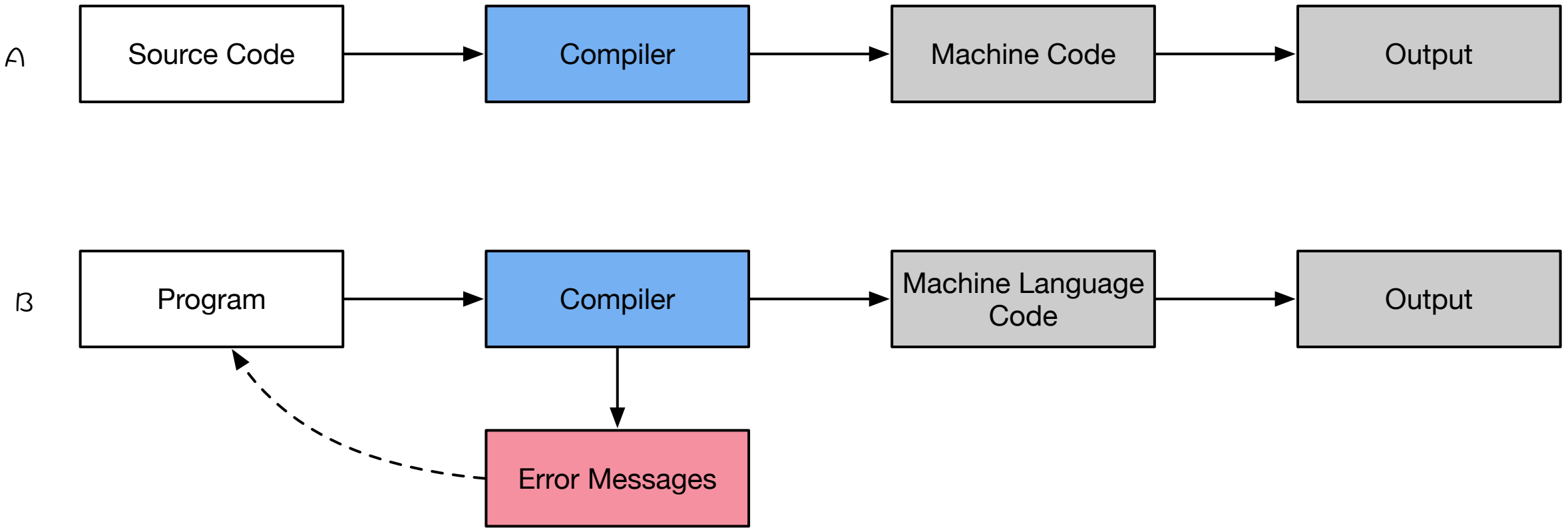
Compiler and Interpreter - 编译器与解释器



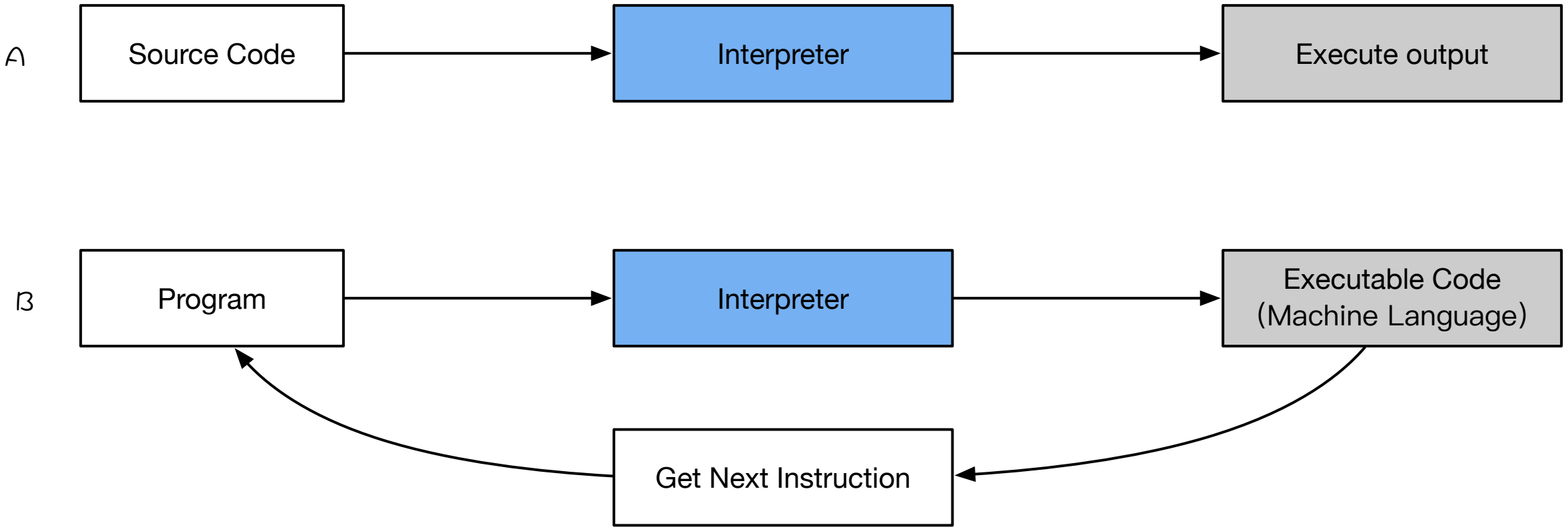
Compiler and Interpreter - 编译器与解释器

	Interpreter 解释器	Compile 编译器
程序步骤	1、创建代码 2、没有文件链接或机器代码生成 3、源语句在执行过程中逐行执行	1、创建代码 2、将解析或分析所有语言语句的正确性 3、将把源代码转换为机器码 4、链接到可运行程序 5、运行程序
Input	每次读取一行	整个程序
Output	不产生任何的中间代码	生成中间目标代码
工作机制	编译和执行同时进行	编译在执行之前完成
存储	不保存任何机器代码	存储编译后的机器代码在机器上
执行	程序执行是解释过程的一部分，因此是逐行执行的	程序执行与编译是分开的，它只在整个输出程序编译后执行
生成程序	不生成输出程序，所以他们在每次执行过程中都要评估源程序	生成可以独立于原始程序运行的输出程序(以exe的形式)
修改	直接修改就可运行	如果需要修改代码，则需要修改源代码，重新编译
运行速度	慢	快
内存	它需要较少的内存，因为它不创建中间对象代码	内存需求更多的是由于目标代码的创建
错误	解释器读取一条语句并显示错误。你必须纠正错误才能解释下一行	编译器在编译时显示所有错误和警告。因此，不修正错误就不能运行程序
错误监测	容易	难
编程语言	PHP, Perl, Python, Ruby	C, C++, C#, Scala, Java

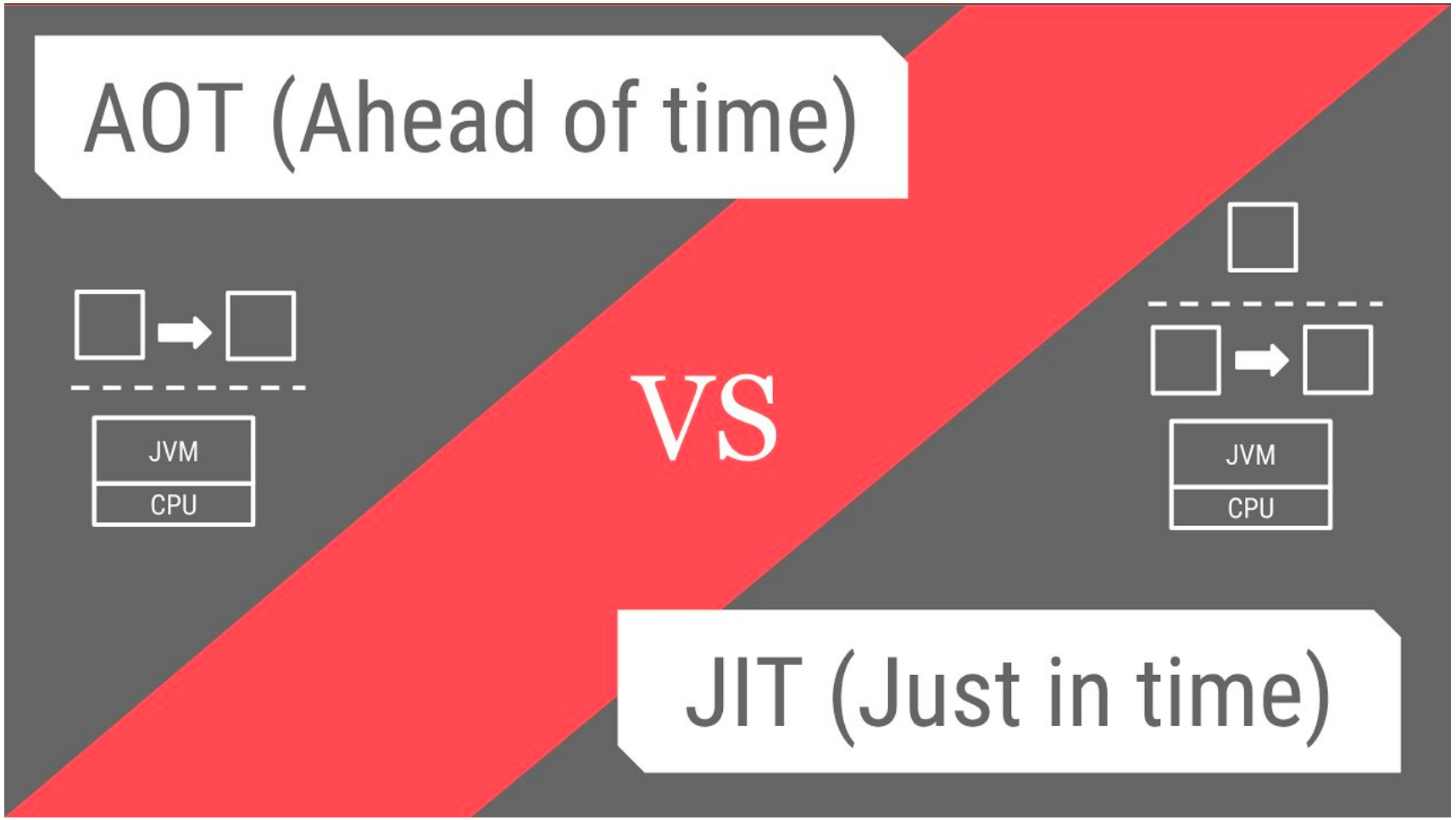
How Compiler Work



How Interpreter Work



Just in time and Ahead of time - JIT和AOT编译方式



Just in time and Ahead of time - JIT和AOT编译方式

程序主要有两种运行方式：

- 静态编译：程序在执行前全部被编译为机器码，称为 AOT（Ahead of time），即“提前编译”；
- 动态解释：程序边编译边运行，通常将这种类型称为 JIT（Just in time），即“即时编译”；

Just in time and Ahead of time - JIT和AOT编译方式

	Just in time	Ahead of time
优点	1. 可以根据当前硬件情况实时编译生成最优机器指令 2. 可以根据当前程序的运行情况生成最优的机器指令序列 3. 当程序需要支持动态链接时，只能使用JIT 4. 可以根据进程中内存的实际情况调整代码，使内存能够更充分的利用	1. 在程序运行前编译，可以避免在运行时的编译性能消耗和内存消耗 2. 可以在程序运行初期就达到最高性能 3. 可以显著的加快程序的启动
缺点	1. 编译需要占用运行时资源，会导致进程卡顿 2. 编译占用运行时间，对某些代码编译优化不能完全支持，需在流畅和时间权衡 3. 在编译准备和识别频繁使用的方法需要占用时间，初始编译不能达到最高性能	1. 在程序运行前编译会使程序安装的时间增加 2. 牺牲高级语言的一致性问题 3. 将提前编译的内容保存会占用更多的外

Question

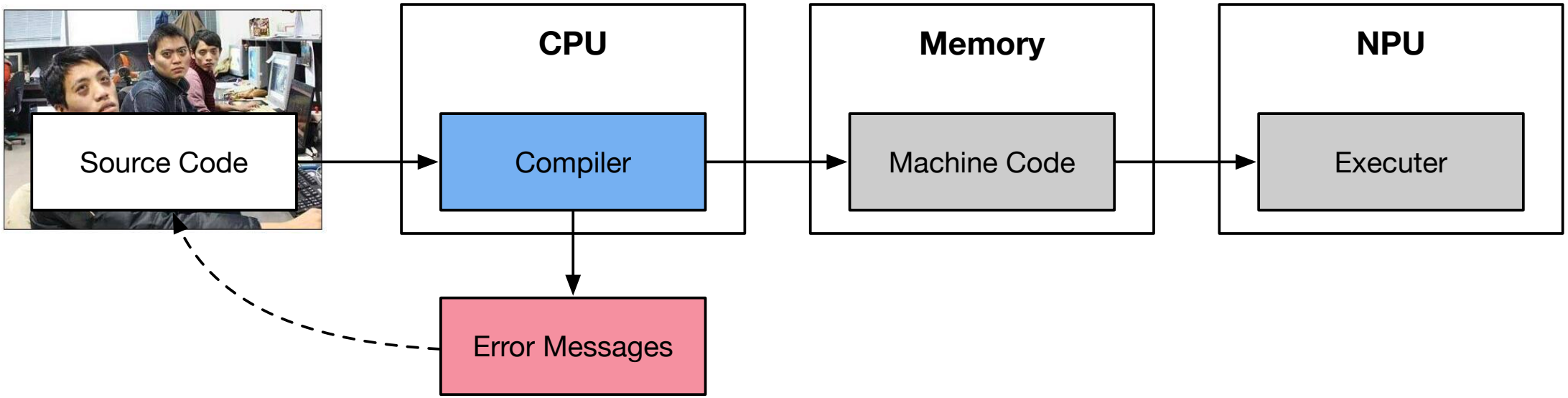
- When did Compiler choice JIT and AOT ?

  什么时候用到 JIT 什么时候用到 AOT ?

Question

- When did Compiler choice JIT and AOT ?

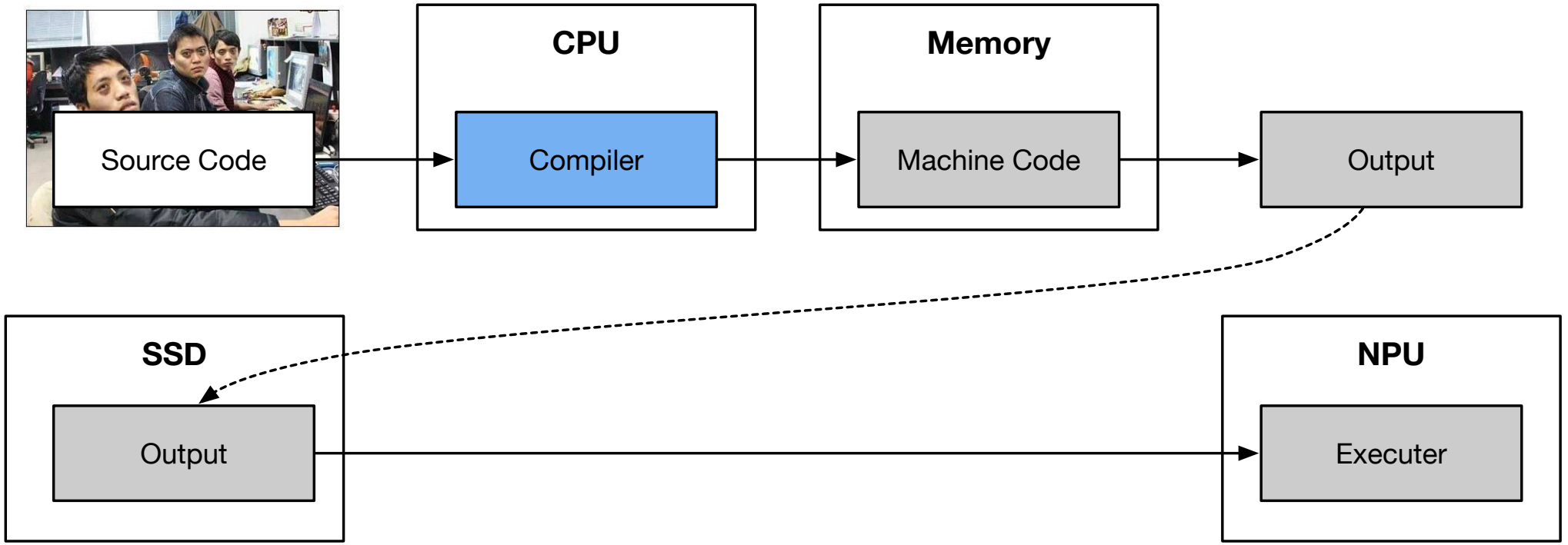
  什么时候用到 JIT 什么时候用到 AOT ?



Question

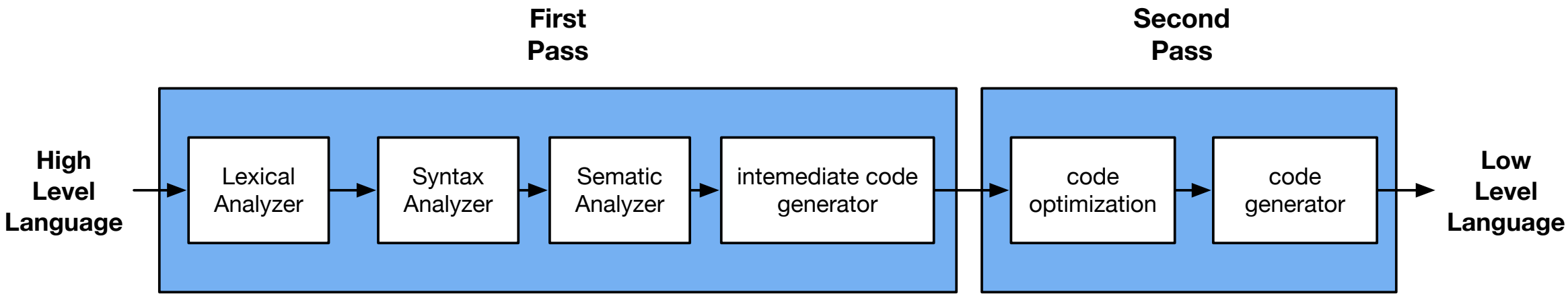
- When did Compiler choice JIT and AOT ?

  什么时候用到 JIT 什么时候用到 AOT ?



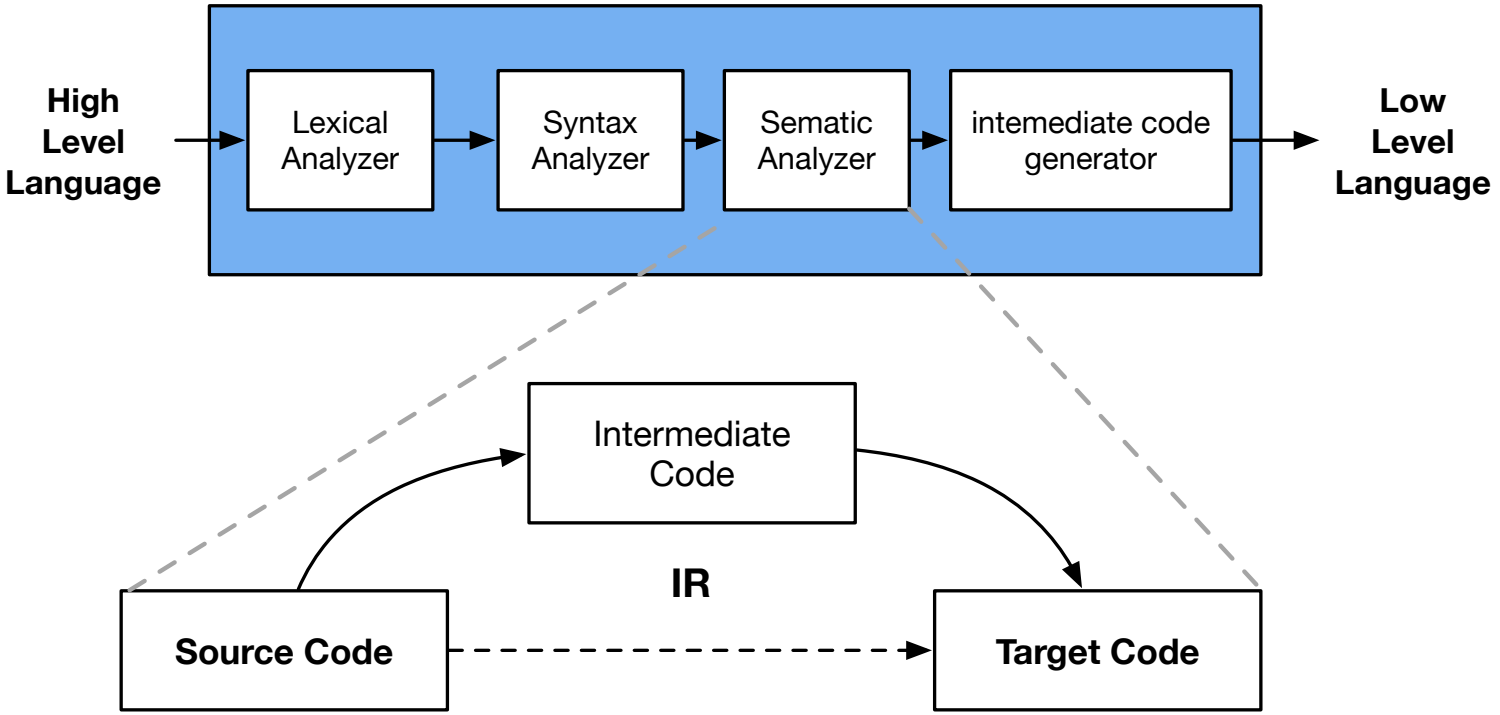
Pass

- Pass：One complete scan or processing of the source program.
  对源程序的一次完整扫描或处理



Intermediate Representations, IR

- IR: An intermediate representation (IR) is the data structure or code used internally by a compiler or virtual machine to represent source code.



Traditional Compiler 传统编译器

AI Compiler - AI编译器