编译原理

何振峰

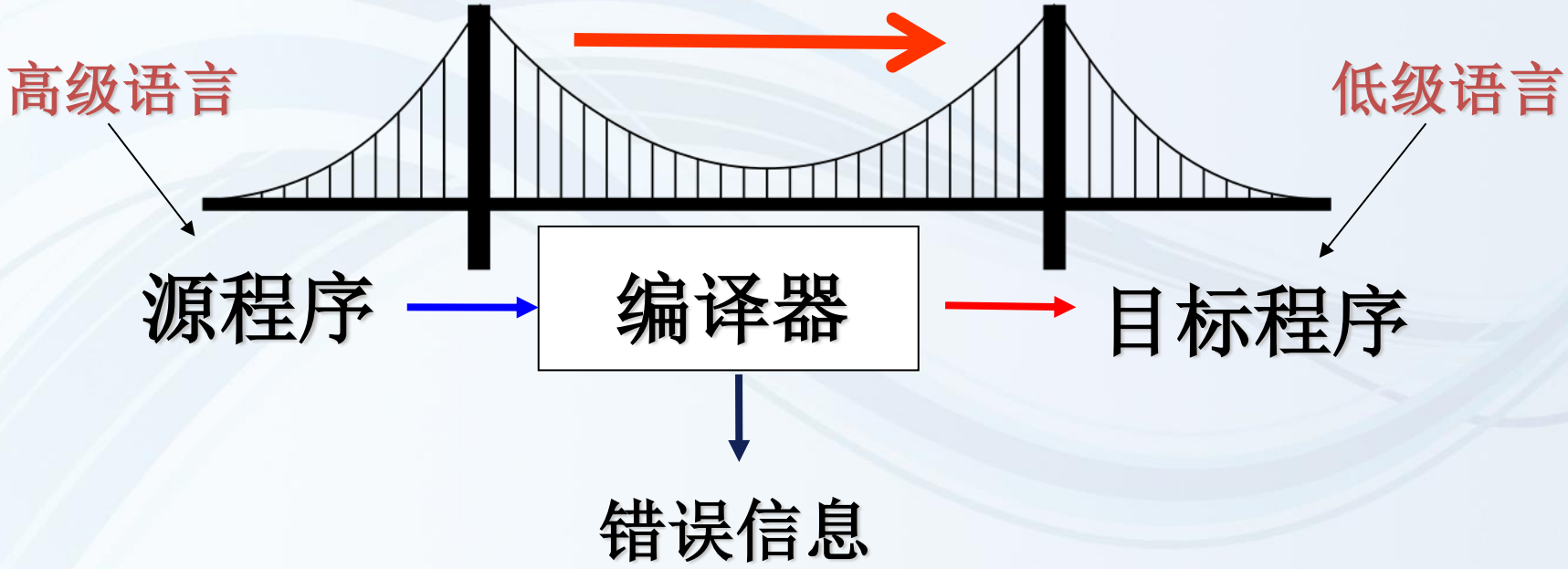
QQ群：1083735727

编译(compile)

- **Compile:** 牛津词典
  - to *produce* a book, list, report, etc. by *bringing together* different items, articles, songs, etc. 编
  - to *translate* instructions from one computer language into another so that a particular computer can *understand* them 译
    - translate: to express the meaning of speech or writing in a different language

1.1语言处理器

- 编译器(compiler)是一个程序，它处理以高级语言（源语言）编写的程序（源程序），将其翻译成等价的、用低级语言（目标语言）编写的程序（目标程序），同时报告源程序中的错误。



其它语言处理器(1)

- 反编译器(decompiler)：进行编译器的反向操作，把程序由较低的抽象形式（机器可读）转换成较高的抽象形式（人工可读）。
  - 源语言为低级语言，目标语言为高级语言。
  - 作用：分析缺少源码的遗留系统（逆向工程）；分析恶意软件
  - 注意：反编译可能违反软件许可协议或知识产权法

其它语言处理器(2)

- 源到源的翻译器（转译器, *transpiler*)：把一种高级语言翻译成为另一种高级语言。
  - 作用：转换遗留系统的代码；
    抽象语言编程

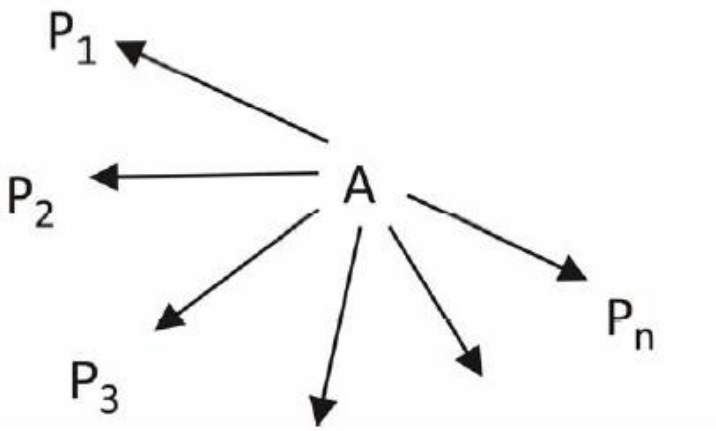
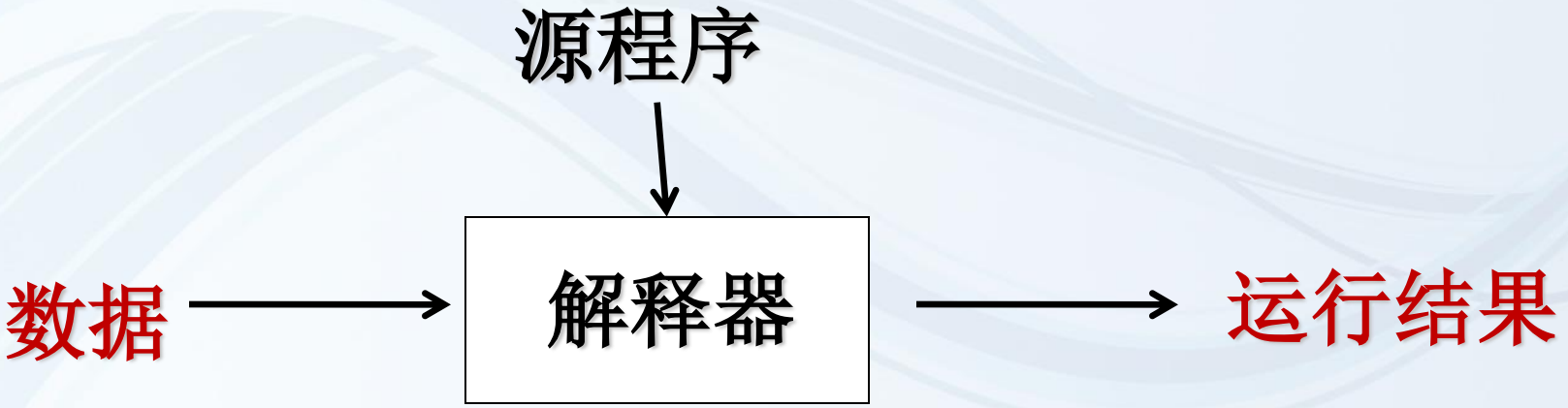


Figure 1. An abstract language.

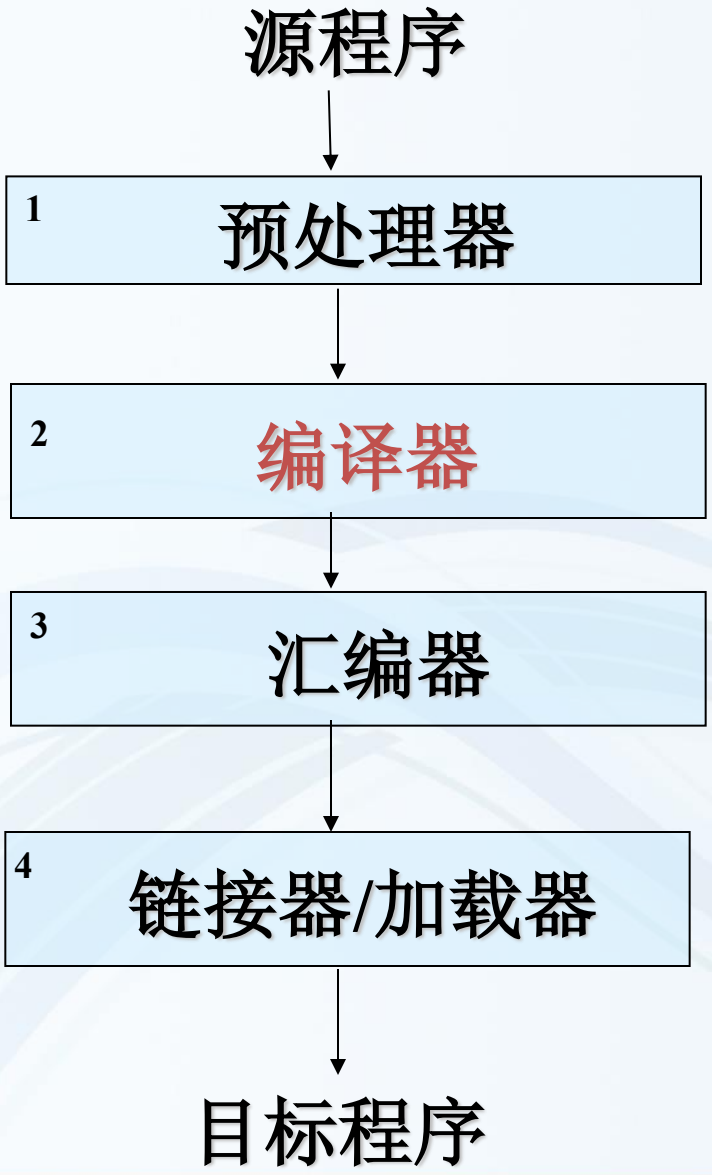
其它语言处理器(3)

❖解释器(*interpreter*)：进行解释并执行输入程序的计算机程序。在线执行

- ➢输入高级语言程序
- ➢不生成目标程序输出



语言处理系统



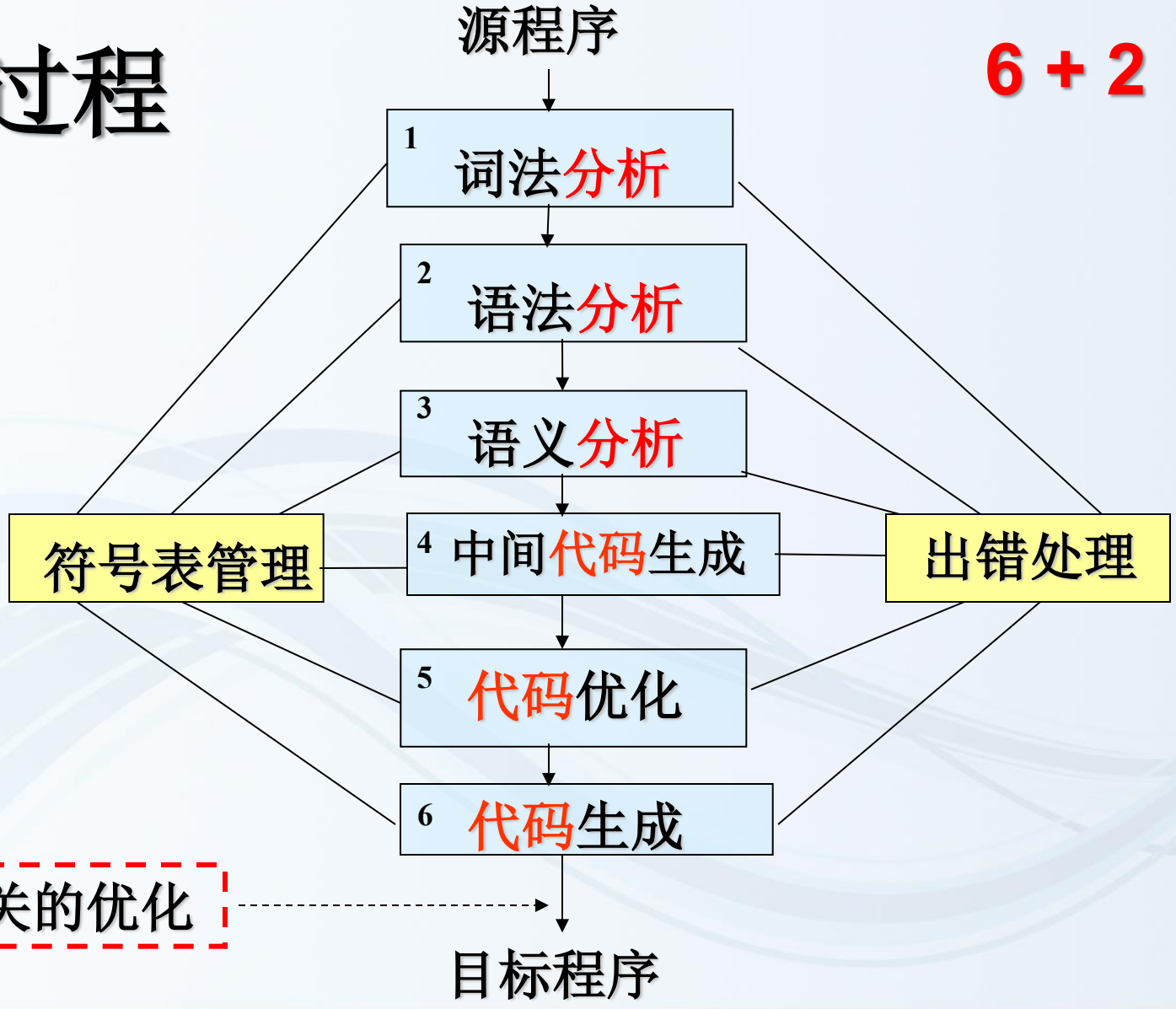
include　　define

1.2 编译器的结构（分析-综合模型）

- 分析
  - 将源程序分解成中间表示
- 综合
  - 根据中间表示和符号表信息来构建目标程序

编译过程

6 + 2


源程序
1 词法分析
2 语法分析
3 语义分析
4 中间代码生成
5 代码优化
6 代码生成
符号表管理
出错处理
机器相关的优化
目标程序

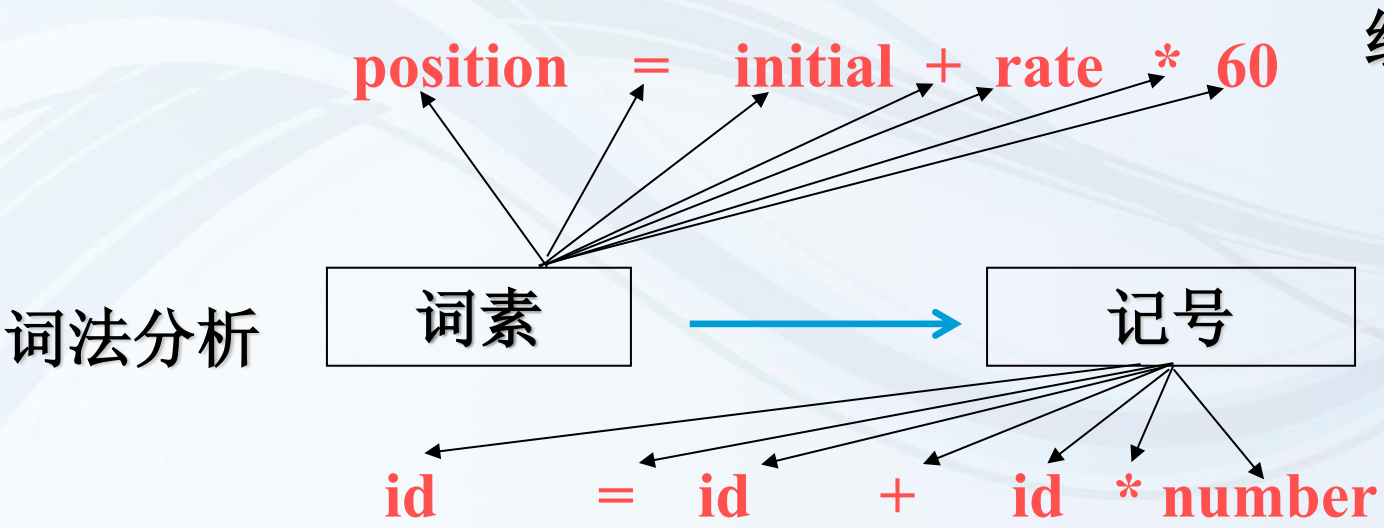
词法分析(*Lexical Analysis*)

❖识别出词素(lexeme)，并抽象成记号(token)

- lexeme: a word or several words that have a meaning that is not expressed by any of its separate parts

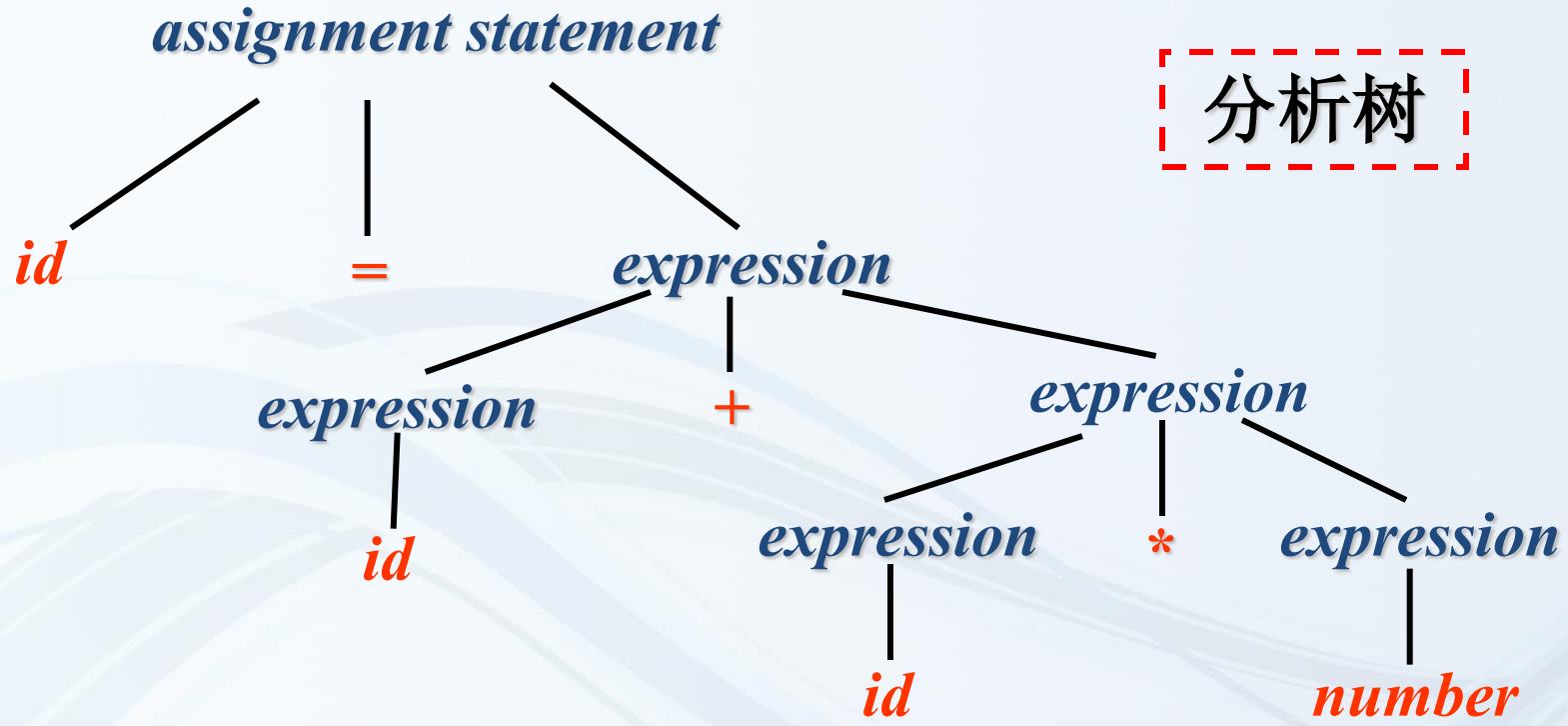
➢记号也称单词或词法单元

例:



空白，换行等将被忽略

语法分析(*Parsing, Syntax Analysis*)

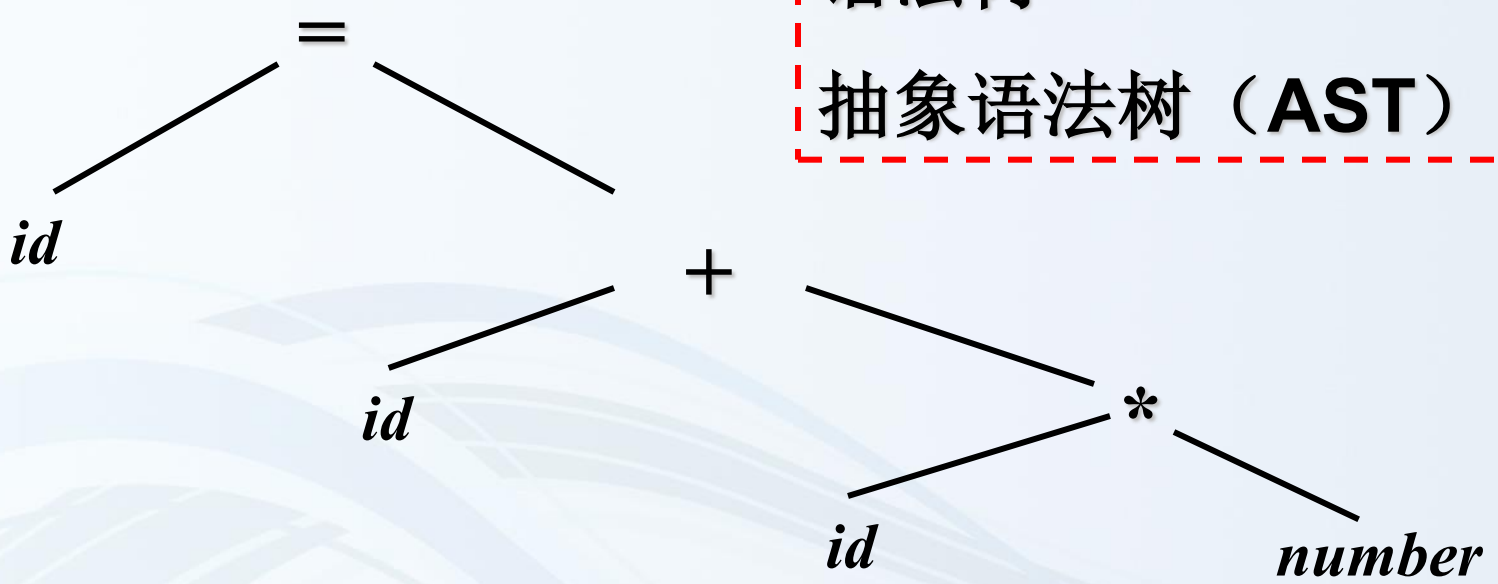


树是基于语言的文法来构建的

语法分析

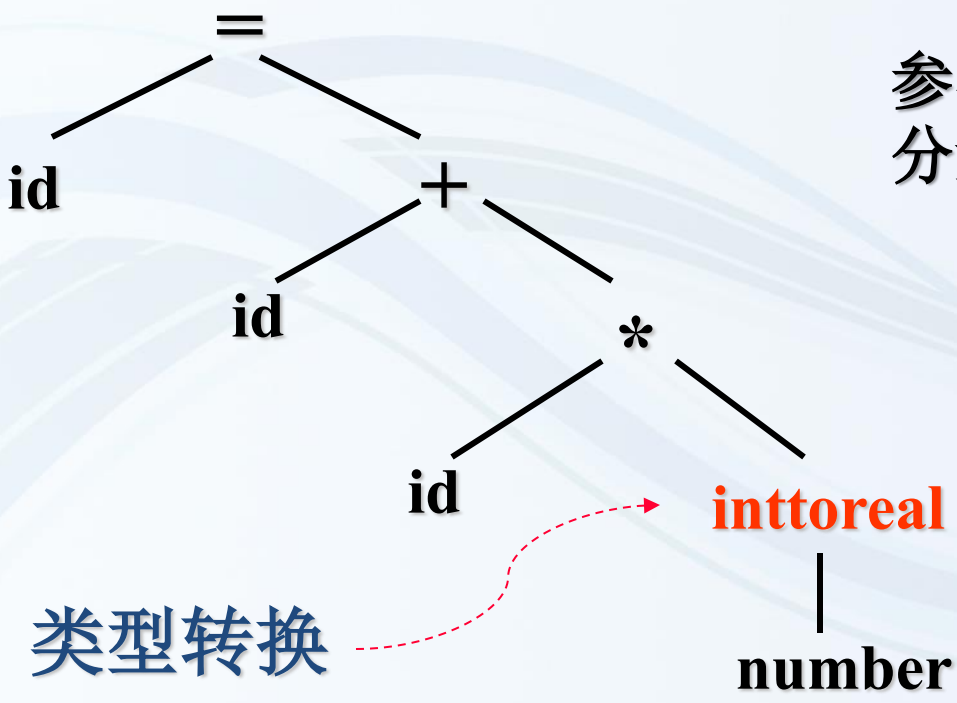
语法树

抽象语法树（AST）



语义分析(*Semantic Analysis*)

- 发现语义错误，并支持代码生成
- 分析树被标记上语义动作



参与运算的两个运算分量的类型应该匹配

语义分析(*Semantic Analysis*)

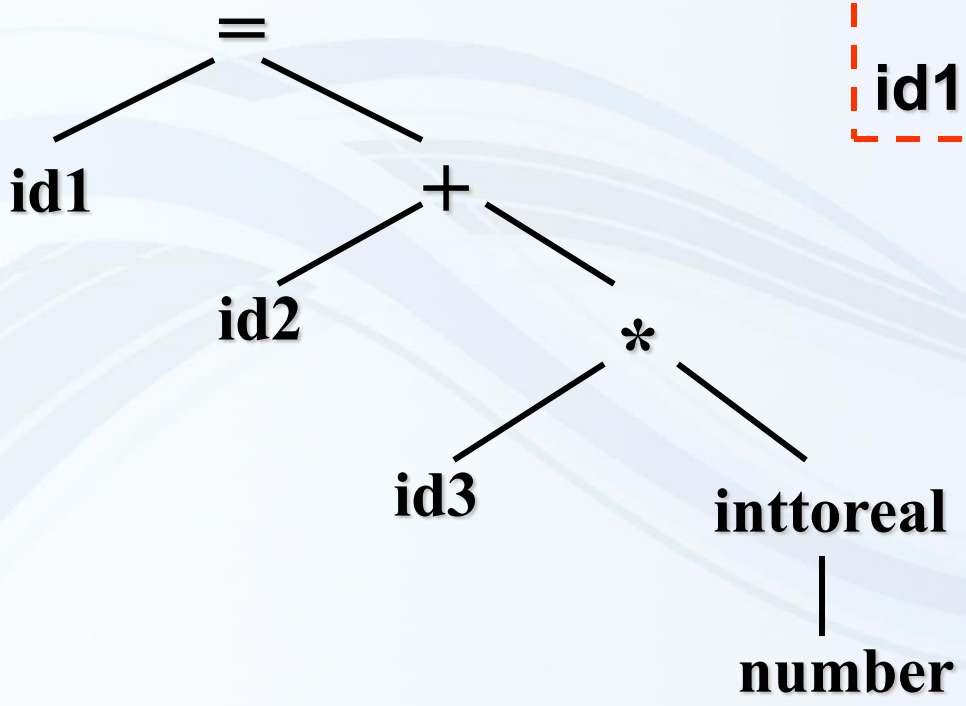
```
template <typename T>
T add(T a, T b) {
    return a + b;
}
```

```
add(3, 5.5);        // 类型 T 应推断为 int 还是 double?
```

中间代码生成

- 三地址码

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```



代码优化

- 将代码转化成更佳的形式
  - 更少的运行时间
  - 更少的空间占用

```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

```
t1 = id3 * 60.0
id1 = id2 + t1
```

代码优化(II)

```
b ← ···
c ← ···
a ← 1

for i = 1 to n
    read d
    a ← a × 2 × b × c × d
    end
```

(a) Original Code in Context

```
b ← ···
c ← ···
a ← 1
t ← 2 × b × c

for i = 1 to n
    read d
    a ← a × d × t
    end
```

(b) Improved Code

目标代码生成

❖产生目标语言代码

```
t1 = id3 * 60.0
id1 = id2 + t1
```

```
LDF   R2,   id3
MULF  R2,   R2,   #60.0
LDF   R1,   id2
ADDF  R1,   R1,   R2
STF   id1,  R1
```

position = initial + rate * 60

- 符号表(symbol table)管理
  - 保存记号的信息（存储空间、类型、作用域）
  - 在分析、综合过程中使用/修改

- 错误处理
  - 不同阶段发现不同的错误
  - 问题
    - ✓发现哪些错误?
    - ✓发现错误后如何处理?

position = initial + rat * 60

position == initial + rate * 60

position = 1nitial + rate * 60

编译阶段的其它划分法

- **前端与后端**
  - 前端与源语言相关，但与目标机器无关，包括：词法分析、语法分析、语义分析、中间代码生成和与目标机器无关的优化。
  - 后端与源语言无关，与目标机器有关，包括：与目标机器有关的优化、目标代码生成。
- **趟**
  - 是对源程序或源程序的中间结果从头到尾扫描一次，进行加工，生产新的中间结果或目标程序。

编译器的重要性（思政进课堂）

- 编译器提供应用和体系结构之间的重要接口，是所有信息系统的要素，影响经济、科技和国家安全。

计算机组成与系统结构
操作系统
汇编语言

C语言
数据库
算法与数据结构
软件工程

TURING AWARD LECTURE

Reflections on Trusting Trust

To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.

Ken Thompson在1983年的图灵奖报告（Reflections on Trusting Trust)中以一个简单的C编译器为例，指出:源于基础软件的安全问题难以避免。

KEN THOMPSON

可以通过编译器恶意植入bug

https://users.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf

```
compile(s)
char *s;
{
        if(match(s, "pattern1")) {
                compile ("bug1");
                return;
        }
        if(match(s, "pattern 2")) {
                compile ("bug 2");
                return;
        }
        ...
}
```

panies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code. In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. A well-installed microcode bug will be almost impossible to detect.

信任具有传递性和递归性

- 我们信任编译器，因为它的源代码看起来正确。
- 但编译器的正确性依赖于另一个编译器（或之前的编译器版本）未被篡改。
- 这种依赖链条可能无限回溯，形成信任黑洞。

可能的解决途径

- **多样化编译（Diverse Compilation）**
  - 使用不同的编译器编译关键代码，降低单一工具链被污染的风险。
- **可验证构建（Reproducible Builds）**
  - 通过多个独立方重复构建二进制文件，验证一致性。
- **形式化验证**
  - 从数学上证明编译器行为的正确性

编译器引入的安全漏洞

- **CISB（*Compiler-Introduced Security Bugs*）**
  - 未优化的代码在目标机器上运行时无安全问题
  - 编译器的优化过程引入了漏洞
  - 该优化从形式上是正确的，没有违背语言要求

```
if(x+10<x){

}
```

可能通过编译器**无意引入**bug

延伸阅读： *Silent bugs matter: a study of compiler-introduced security bugs*
https://www.usenix.org/system/files/sec23fall-prepub-123-xu-jianhao.pdf

构建自主可控、兼容开放的基础软硬件生态体系是实现信息产业健康发展的保障

❖ 自主可控（Independence and Controllability）

由自己主导设计、研制、生产、维护，关键技术不受制于他人，能够保证产品的安全性、使用的可靠性、供货的稳定性。

❖ 基础软硬件生态体系

由硬件（芯片、存储等）、系统软件（操作系统、中间件）、开发工具链（编译器、调试器等）、应用软件及开发者社区构成的综合技术体系，其核心目标是实现从底层硬件到上层应用的完整技术闭环。

常见错误认识1

- ❖错误观点：美国对中国的禁运，始于特朗普时代
- ❖事实：从巴黎统筹到芯片法案
  - ➢巴黎统筹委员会（COCOM，美国主导）：1949-1994

    中国委员会成立于1952年，对中国进行最彻底的技术封锁
  - ➢瓦森纳协定（美国主导）：1996至今
  - ➢沃尔夫条款（美国单独，全球实施）：2011至今

    禁止中美之间的航天合作
  - ➢芯片与科学法案（美国单独，全球实施）：2022年8月至今

    限制在中国投资28纳米以下制程的芯片技术
  - ➢美国商务部、联邦通信委员会、国防部、总统行政令黑名单

    制裁中国信息产业相关的大多数龙头企业和部分高校

长期以来，以美国为代表的一些西方国家，奉行和实施对中国的货物、服务和技术的出口管制和禁运措施，极大损害了我国的国家利益、产业利益和广大消费者的权益。特别是当我国科技人员自主研发出被禁运装备、技术及软件等以后，美国和其他国家往往立即解禁该类产品与技术，向我国大肆倾销。

节选自：2015年3月《中国计算机学会关于制定反禁运法的建议》

延伸阅读： 对中国禁运的那些先进设备和技术（2017-03-18）

https://www.sohu.com/a/129273893_132567

常见错误认识2

- ❖错误观点：禁运着力芯片，是因为中国硬件更弱
- ❖事实：中国的硬件产业相对于基础软件更有竞争力

延伸阅读

top 20 software companies https://ictbuz.com/software-companies/

top 10 software companies
https://www.investopedia.com/articles/personal-finance/121714/worlds-top-10-software-companies.asp

These States Have Banned DeepSeek

https://statetechmagazine.com/article/2025/04/these-states-have-banned-deepseek

常见错误认识3

- ❖错误观点：禁运着力芯片，是因为软件无法制裁
- ❖事实：对于软件产业的制裁刚刚开始

延伸阅读：

我国高校（哈尔滨工业大学等）被美国禁用Matlab，一场“战争”悄然开始

https://www.163.com/dy/article/FFI7FO6A0545B3M0.html

Microsoft suspends new sales in Russia

https://blogs.microsoft.com/on-the-issues/2022/03/04/microsoft-suspends-russia-sales-ukraine-conflict/

参考书

- 王生原、董渊、张素琴、吕映芝、蒋维杜.编译原理（第3版），北京：清华大学出版社，2015
- 张幸儿. 计算机编译原理（第3版），北京：科学出版社，2015
- 郭旭 译. 编译器设计（第2版），北京：人民邮电出版社，2022

在线资源

- 哈工大编译课程(2026年3月2日 ~ 2026年07月31日)

  第1讲-第14讲

  https://www.icourse163.org/course/HIT-1002123007

- 国防科大编译课程(2026年3月2日 ~ 2026年7月12日)

  第1讲-第19讲（不含8.3,10.4-10.6)

  https://www.icourse163.org/course/NUDT-1003101005

- deepseek

  https://www.deepseek.com/