# 第一章　引论

廖力
xobjects@seu.edu.cn
3793235

# 1.1程序设计语言与编译

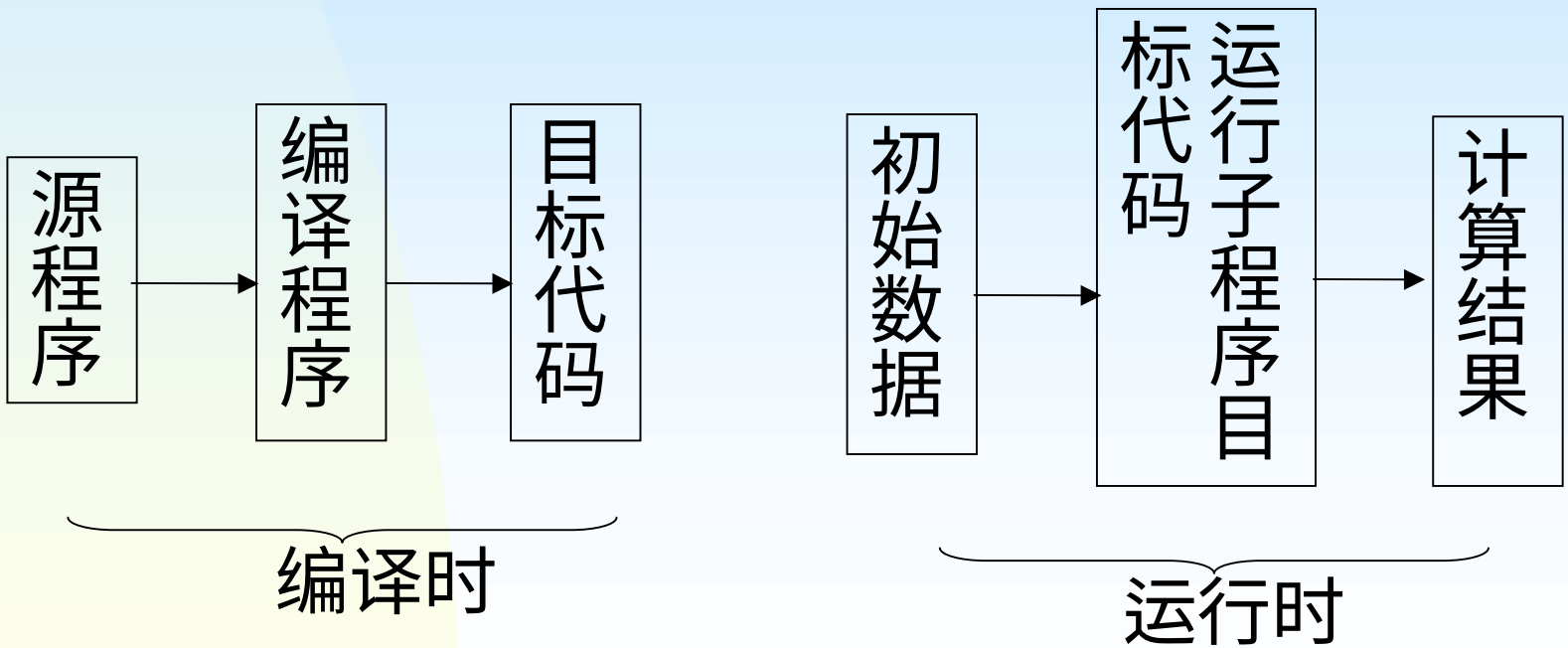
- 1)程序设计语言
  - 高级语言
  - 汇编语言
  - 机器语言
- 在计算机上如何执行一个高级语言程序？
  - 把高级语言程序翻译成机器语言程序
  - 运行所得的机器语言程序求得计算结果

# 1.1程序设计语言与编译
# 2)程序设计语言的转换

- 翻译
  - 是指能把某种语言的源程序，在不改变语义的条件下，转换成另一种语言程序——目标语言程序。
- 编译
  - 专指由高级语言转换为低级语言
- 解释
  - 接受某高级语言的一个语句输入，进行解释并控制计算机执行，马上得到这句的执行结果，然后再接受下一句。

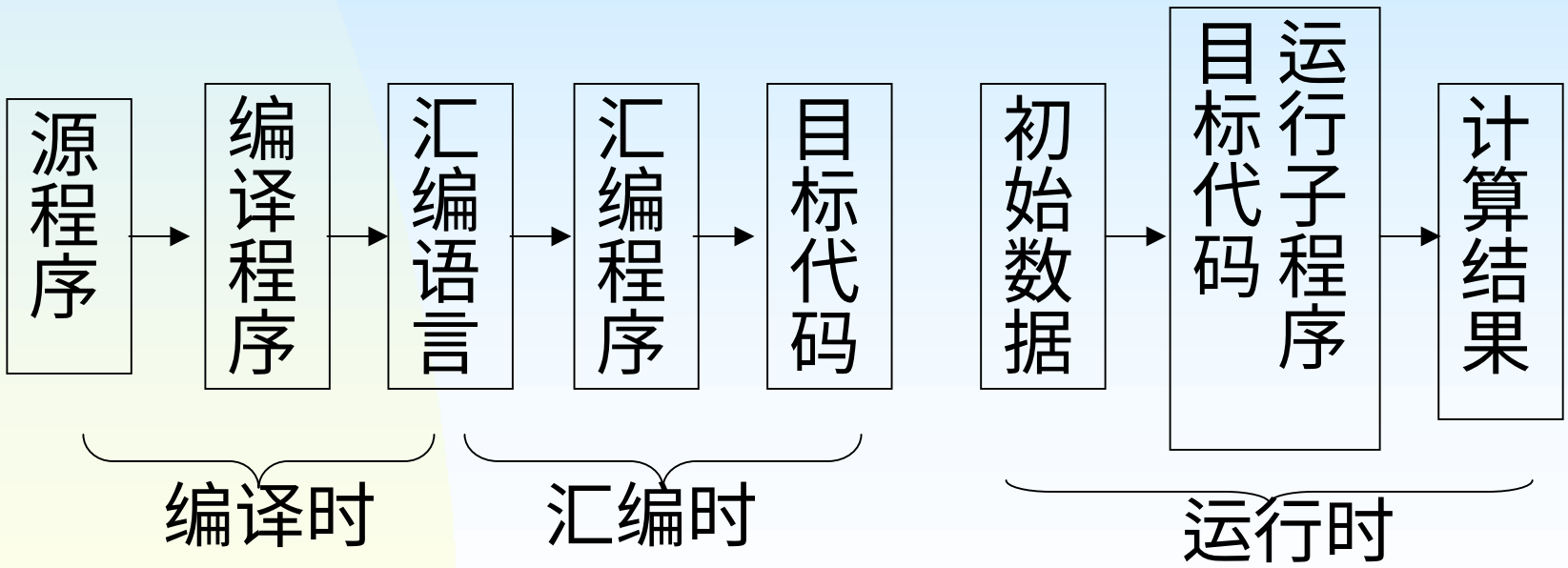
# 1.1程序设计语言与编译
# 2)程序设计语言的转换（续）

- 编译的转换过程
  - 两阶段转换：编译——运行



# 1.1程序设计语言与编译
# 2)程序设计语言的转换（续）

- 编译的转换过程
  - 三个阶段的转换：编译——汇编——运行



# 1.1程序设计语言与编译
# 2)程序设计语言的转换（续）

- 翻译
- 编译
- 解释
  - 以源程序作为输入，不产生目标程序，一边解释一边执行。
  - 优点：直观易懂，结构简单，易于实现人机对话
  - 缺点：效率低

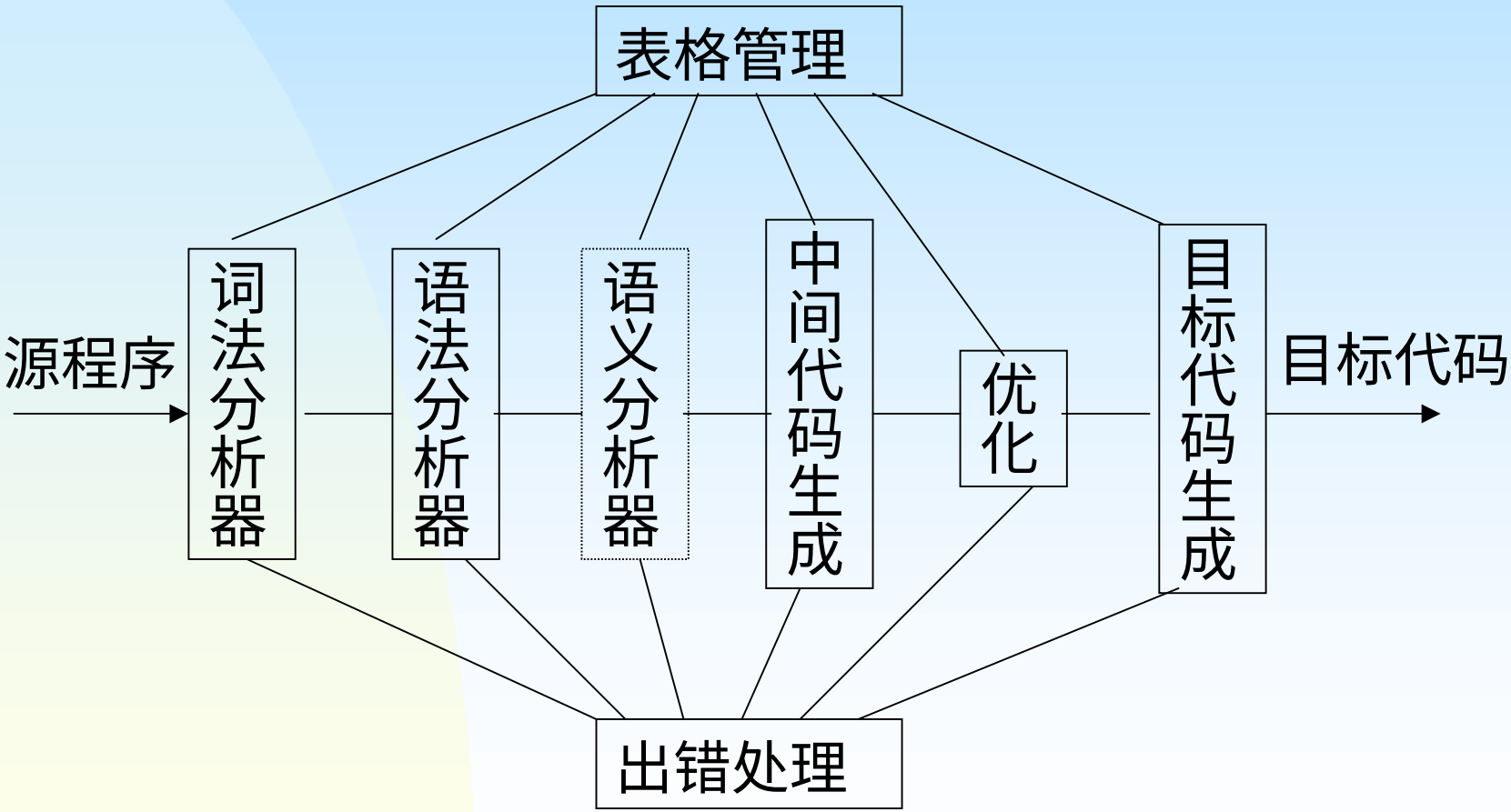
# 1.2编译程序概述

- 编译程序的工作
- 先看自然语言的翻译
  - 1.识别出句子中的一个个单词
  - 2.分析句子的语法结构
  - 3.根据句子的含义进行初步翻译
  - 4.对译文进行修饰
  - 5.写出最后译文

# 1.2编译程序概述
# 编译程序的工作

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 优化
- 目标代码生成

# 编译程序的工作



# 1.2编译程序概述
## 1.词法分析

- 任务
  - 输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个的单词。
- 单词
  - 是高级语言中有实在意义的最小语法单位，它由字符构成。

# 识别右边程序中的单词

- 基本字：

  Void , int , float

- 标识符

  a, b, c, d, x, y,

  jisuan

- 整常数 ： 50
- 运算符 ： +,-,*,=
- 界限符:

  { } ; , ( )

```
Void   jisuan()
{int y,c,d;
   float x,a,b;
   x=a+b*50;
   y=c+)d*(x+b;
   }
```

# 1.2编译程序概述
# 1.词法分析（续）

- 词法分析依照词法规则，识别出正确的单词，转换成统一规格，备用。
- 转换
  - 对基本字、运算符、界限符的转换
  - 标识符的转换
  - 常数的转换
  - 转换完成后的格式：（类号、内码）
- 描述词法规则的有效工具是正规式和有限自动机

# 1.2编译程序概述
# 编译程序的工作

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 优化
- 目标代码生成

# 1.2编译程序概述
# 2.语法分析

- 任务：
  - 在词法分析的基础上，根据语言的语法规则，把单词符号组成各类的语法单位：短语、子句、语句、过程、程序。
- 语法规则：
  - 语言的规则，又称为文法；规定单词如何构成短语、语句、过程和程序。
- 语法规则的表示：
  - BNF：　A::=B|C

# 赋值语句的语法规则

- A::=V=E
- E::=T|E+T
- T::=F|T*F
- F::=V|(E)|C
- V::=标识符
- C::=常数

# 1.2编译程序概述
# 2.语法分析（续）

- 语法分析的方法
  - 推导（derive)和归约(reduce)
- 推导
  - 最左推导、最右推导
- 归约
  - 最右归约、最左归约

# 最右推导，最左归约

- A⇒V=E ⇒ V=E+T ⇒V=E+T*F ⇒V=E+T*C ⇒V=E+T*50
  ⇒V=E+F*50 ⇒V=E+V*50 ⇒V=E+b*50 ⇒V=T+b*50
  ⇒V=F+b*50 ⇒V=V+b*50 ⇒V=a+b*50

⇒x=a+b*50

- 参见书P4文法

# 最左推导，最右归约

- A⇒V=E ⇒ x=E ⇒x=E+T ⇒x=T+T ⇒x=F+T ⇒x=V+T
  ⇒x=a+T ⇒x=a+T*F ⇒x=a+F*F ⇒x=a+V*F
  ⇒x=a+b*F ⇒x=a+b*C
- ⇒x=a+b*50
- 参见书P4文法

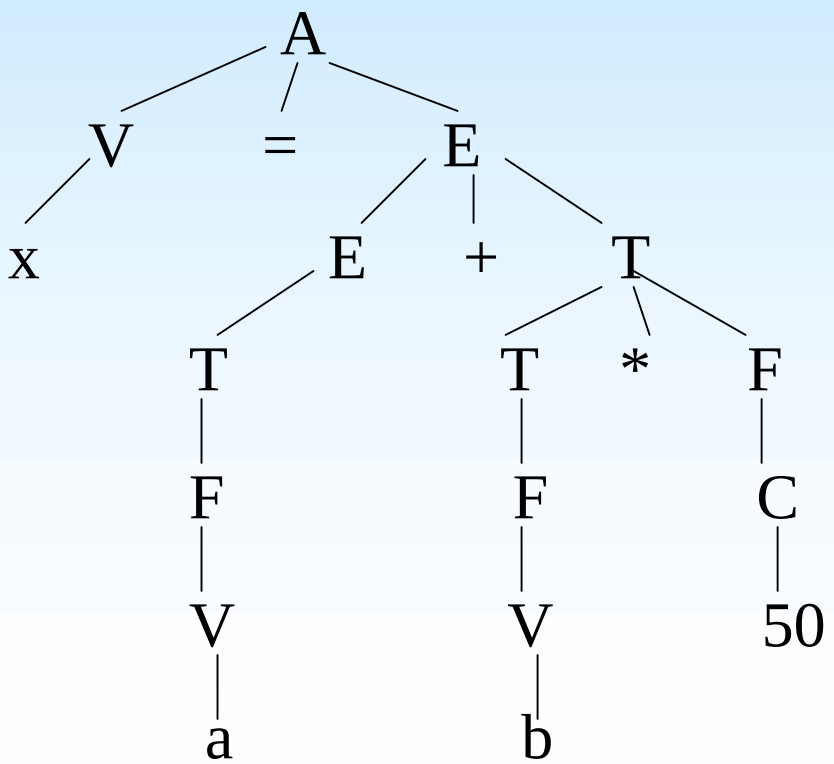
# 再如：

- C语言语句 y=c+)d*(x+b
- 分析过程(最右推导)
- A⇒V=E ⇒V=E+T ⇒V=E+F ⇒V=E+V ⇒V=E+b
  ⇒V=T+b ⇒V=T*F+b ⇒V=T*V+b ⇒V=T*x+b
- 无法得到该语句
- 故，该C语言语句的语法是错误的。

# 语法分析的方法（续）

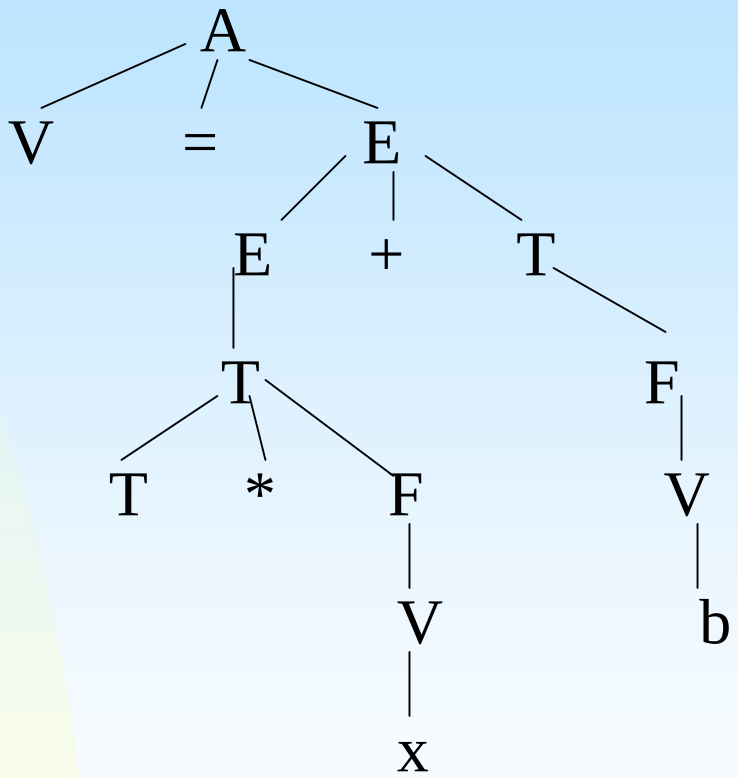
- 语法分析过程也可以用一棵倒着的树来表示
- 这棵树叫做语法树

Eg:x=a+b*50的语法树



# 语法分析的方法（续）

- 再如y=c+)d*(x+b的语法树




注：语法分析对说明语句的处理主要是填符号表，对一般语句的处理是构造语法树。

# 1.2编译程序概述
# 编译程序的工作

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 优化
- 目标代码生成

# 3.中间代码生成

- 任务

  对语法分析识别出的各类语法范畴，分析其含义，进行和初步翻译，产生介于源代码和目标代码之间的一种代码。

- 分为两阶段工作
  - 对每种语法范畴进行静态语义检查
  - 若语义正确，就进行中间代码的翻译
- 中间代码形式
  - 四元式、三元式、逆波兰式

# 中间代码生成（续）

- 例如将x=a+b*50变成中间代码

| 序号 | 算符 | 左操作数 | 右操作数 | 结果 |
|---|---|---|---|---|
| (1) | 将整常数50转换为实常数 | | | $T_1$ |
| (2) | * | b | $T_1$ | $T_2$ |
| (3) | + | a | $T_2$ | $T_3$ |
| (4) | = | $T_3$ | | x |

# 1.2编译程序概述
# 编译程序的工作

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 优化
- 目标代码生成

# 4.优化

- 任务

  对前面产生的中间代码进行加工变换，以期在最后阶段能产生更为高效的目标代码。
- 原则：等价变换
- 主要方面

  公共子表达式的提取、合并已知量、删除无用语句、循环优化等。

# 优化（续）

- 例如将下面的语句转换成中间代码
- For (k=1;k<=100;k++)
- {m=i+10*k;
- n=j+10*k;
- }

```
      K=1;
10    If  k<=100  then
      {m=i+10*k;
       n=j+10*k;
       k++;
       goto 10;}
```

| 序号 | OP | ARG1 | ARG2 | RESULT |
|---|---|---|---|---|
| (1) | = | 1 | | K |
| (2) | j< | 100 | K | (9) |
| (3) | * | 10 | K | $T_1$ |
| (4) | + | i | $T_1$ | M |
| (5) | * | 10 | K | $T_2$ |
| (6) | + | j | $T_2$ | N |
| (7) | + | k | 1 | K |
| (8) | j | | | (2) |
| (9) | | | | |

```
      K=1;
10    If k<=100
then
      {m=i+10*k;
       n=j+10*k;
       k++;
       goto 10;}
```

| 序号 | OP | ARG1 | ARG2 | RESULT |
|---|---|---|---|---|
| (1) | = | i | | m |
| (2) | = | j | | n |
| (3) | = | 1 | | k |
| (4) | J< | 100 | k | (9) |
| (5) | + | m | 10 | m |
| (6) | + | n | 10 | n |
| (7) | + | k | 1 | k |
| (8) | j | | | (4) |
| (9) | | | | |

# 1.2编译程序概述
# 编译程序的工作

- 词法分析
- 语法分析
- 语义分析和中间代码生成
- 优化
- 目标代码生成

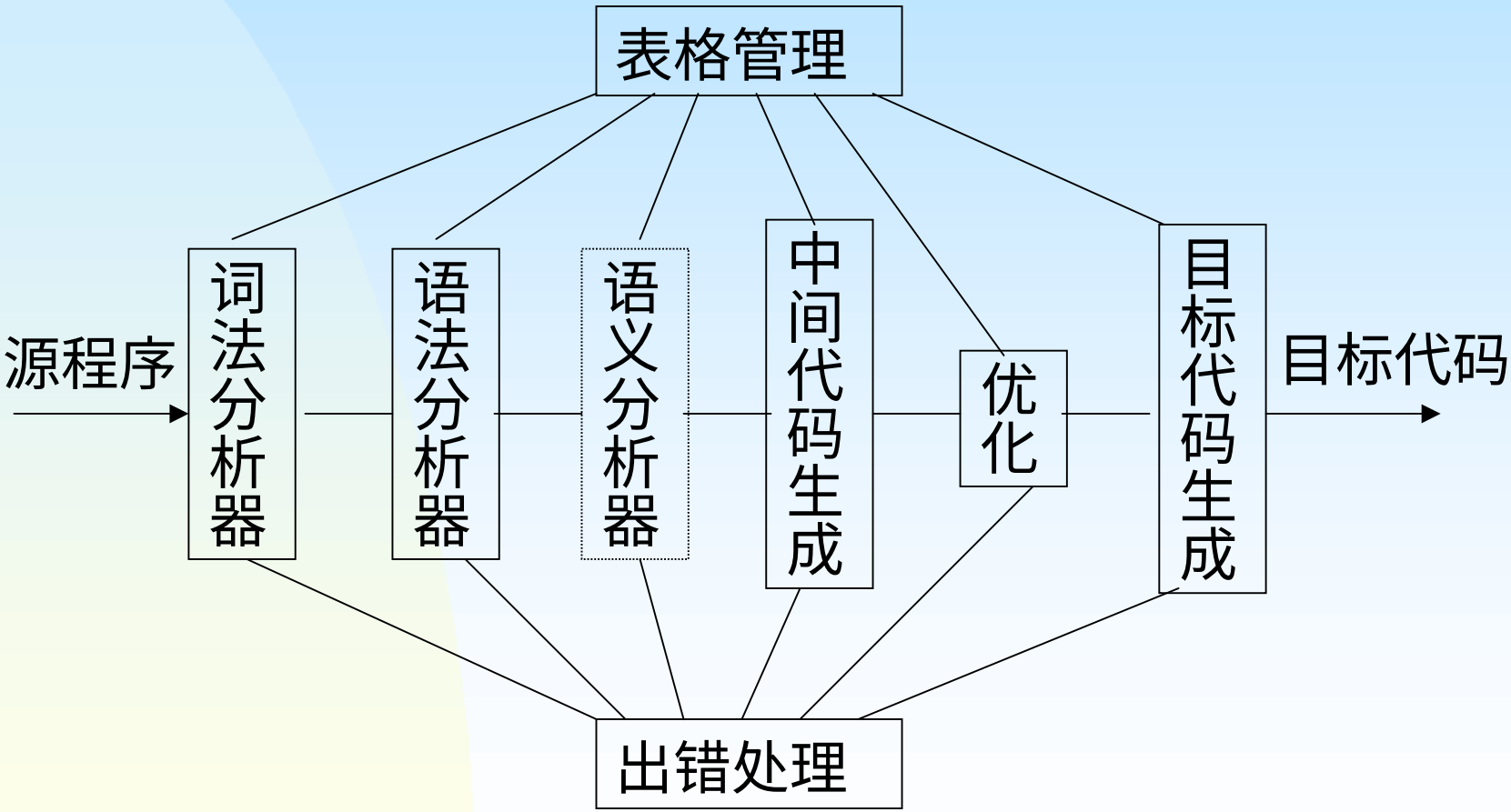
# 5.目标代码生成

- 任务

  把经过优化的中间代码转化成特定机器上的低级语言代码。

- 目标代码的形式
  - 绝对指令代码:可立即执行的目标代码。
  - 汇编指令代码：汇编语言程序，需要通过汇编程序汇编后才能运行。
  - 可重定位指令代码：先将各目标模块连接起来，确定变量、常数在主存中的位置，装入主存后才能成为可以运行的绝对指令代码。

# 编译程序的工作



# 6.表格与表格管理

- 表格作用：
  - 用来记录源程序的各种信息以及编译过程中的各种状况。
- 与编译前三阶段有关的表格有：
  - 符号表、常数表、标号表、分程序入口表、中间代码表等。

# 1）符号表

- 符号表：用来登记源程序中的常量名、变量名、数组名、过程名等，记录它们的性质、定义和引用情况。

| NAME | INFORMATION |
| --- | --- |
| m | 整型、变量地址 |
| n | 整型、变量地址 |
| k | 整型、变量地址 |

# 2）常数表与标号表

常数表

| 值 |
| --- |
| 1 |
| 4 |

（登记各类常量值）

标号表

| NAME | INFORMATION |
| --- | --- |
| …… | …… |
| 10 | 四元式序号4 |

（登记标号的定义与应用）

# 3）入口名表

- 作用：登记过程的层号，分程序符号表入口等

| NAME | INFORMATION |
| --- | --- |
| …… | ……. |
| INCWAP | 二目子程序、四元式序号1 |

# 4）中间代码表

| 序号 | OP | ARG1 | ARG2 | RESULT |
|---|---|---|---|---|
| (1) | = | i | | m |
| (2) | = | j | | n |
| (3) | = | 1 | | k |
| (4) | J< | 100 | k | (9) |
| (5) | + | m | 10 | m |
| (6) | + | n | 10 | n |
| (7) | + | k | 1 | k |
| (8) | j | | | (4) |
| (9) | return | | | |

# 7.出错处理

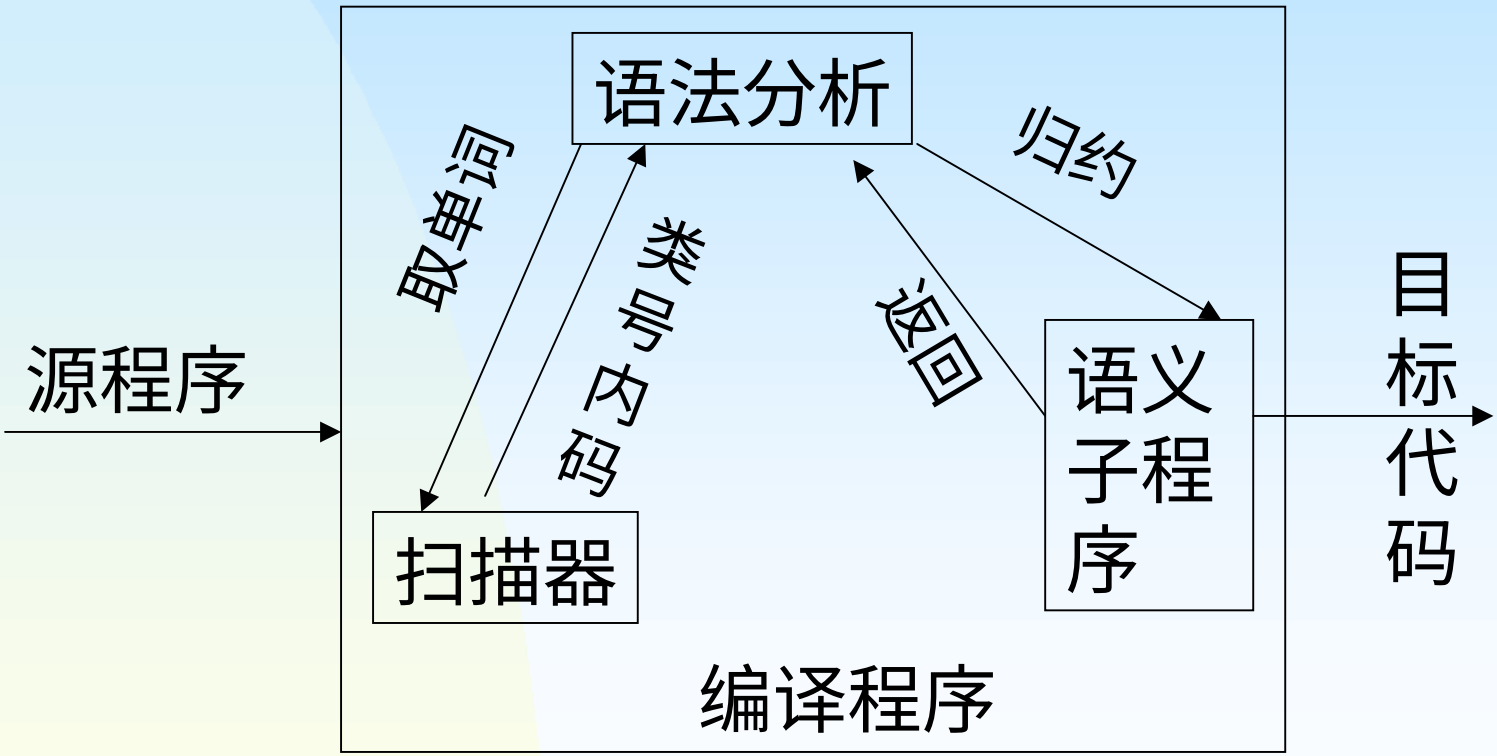
- 任务

  如果源程序有错误，编译程序应设法发现错误，并报告给用户。
- 完成：由专门的出错处理程序来完成
- 错误类型:
  - 语法错误:在词法分析和语法分析阶段检测出来。
  - 语义错误：一般在语义分析阶段检测。

# 8.遍

- 遍：指对源程序或源程序的中间结果从头到尾扫描一次，并做有关的加工处理，生成新的中间结果或目标代码的过程。
  - 注：遍与阶段的含义毫无关系。
- 一遍扫描
- 多遍扫描
  - 优点：节省内存空间，提高目标代码质量，使编译的逻辑结构清晰。
  - 缺点：编译时间较长。
  - 注：在内存许可情况下，还是遍数尽可能少些为好。

# 一遍扫描（以语法分析为中心）



# 1.3编译程序生成

- 1.直接用机器语言编写编译程序
- 2.用汇编语言编写编译程序
  - 注：编译程序核心部分常用汇编语言编写
- 3.用高级语言编写编译程序
  - 注：这是普遍采用的方法

# 1.3编译程序生成（续）

- 4.自编译
- 5.编译工具:
  - LEX(词法分析)与YACC(用于自动产生LALR分析表)
- 6.移植（同种语言的编译程序在不同类型的机器之间移植）

# 1.4编译程序构造

- 在某机器上为某种语言构造编译程序要掌握以下三方面：
  - 源语言
  - 目标语言
  - 编译方法

# 本章小结

- 掌握编译程序与高级程序设计语言的关系；
- 掌握编译分为哪几个阶段；
- 了解各个阶段完成的主要功能和采用的主要方法。