第一章 引论

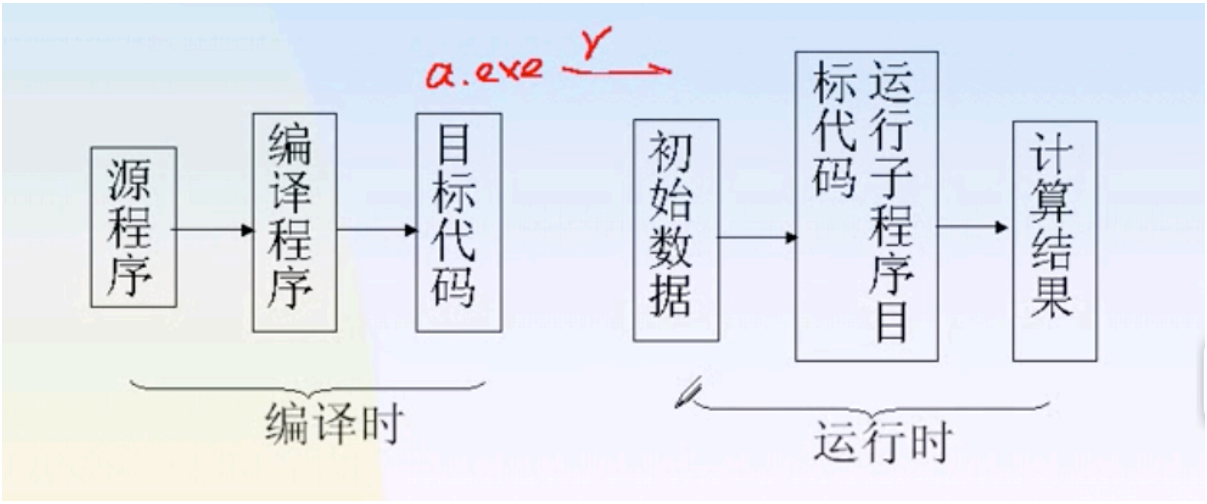
编译原理：将高级程序设计语言变换成计算机硬件所能识别的机器语言，以便计算机进行处理。

程序设计语言：
1.高级语言
2.汇编语言(汇编语言就是机器语言的抽象，非常接近机器级，要针对某个机器而言，因为每种机器的机器语言不一样，很不方便)
3.机器语言(机器只能识别0和1)

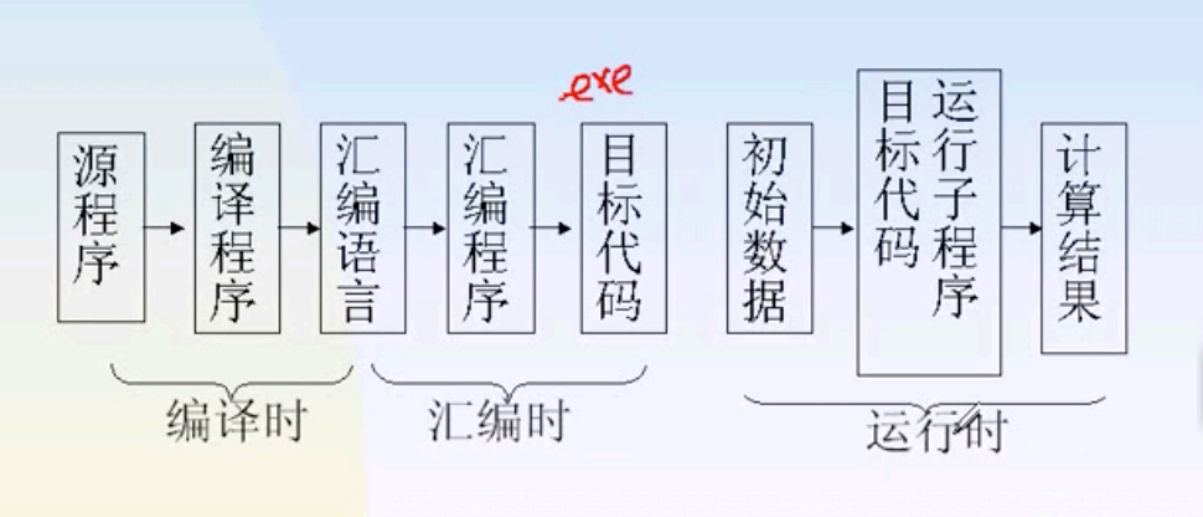
计算机如何执行一个高级语言程序?
1.把高级语言程序翻译成机器语言程序
2.运行所得的机器语言程序求得计算结果

翻译：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即目标语言程序。
翻译有两种：
编译：专指由高级语言转换成低级语言
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。

编译的转换过程：两个阶段的转换：编译—>运行



编译的转换过程：三个阶段的转换：编译—>汇编—>运行
(区别在于编译出的结果不一样)



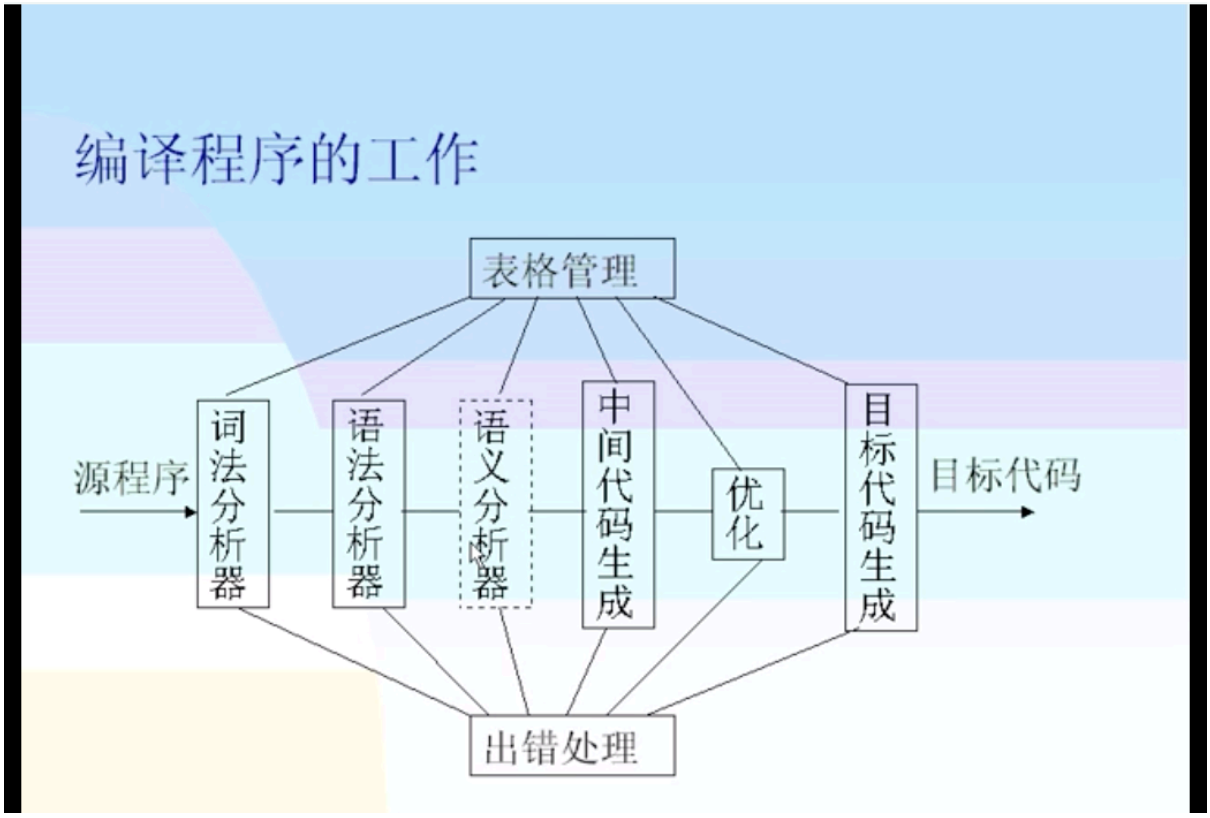
[目标代码可能是exe，也可能是obj。obj要经历link过程]

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。【效率低的原因：不产生目标文件，每次都得重新解释】

编译程序概述：
可以参考自然语言的翻译。编译也就是高级语言到低级语言的翻译。
编译程序的工作：
1.词法分析【分析单词写的对不对】
2.语法分析【看一看这个句子起什么作用，是判断赋值or循环...？？】
3.语义分析和中间代码生成【这句话意思对不对，中间代码是基于句子和01代码之间】
4.优化
5.目标代码生成

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。

1.词法分析
2.语法分析
3.语义分析和中间代码生成
4.优化
5.目标代码生成



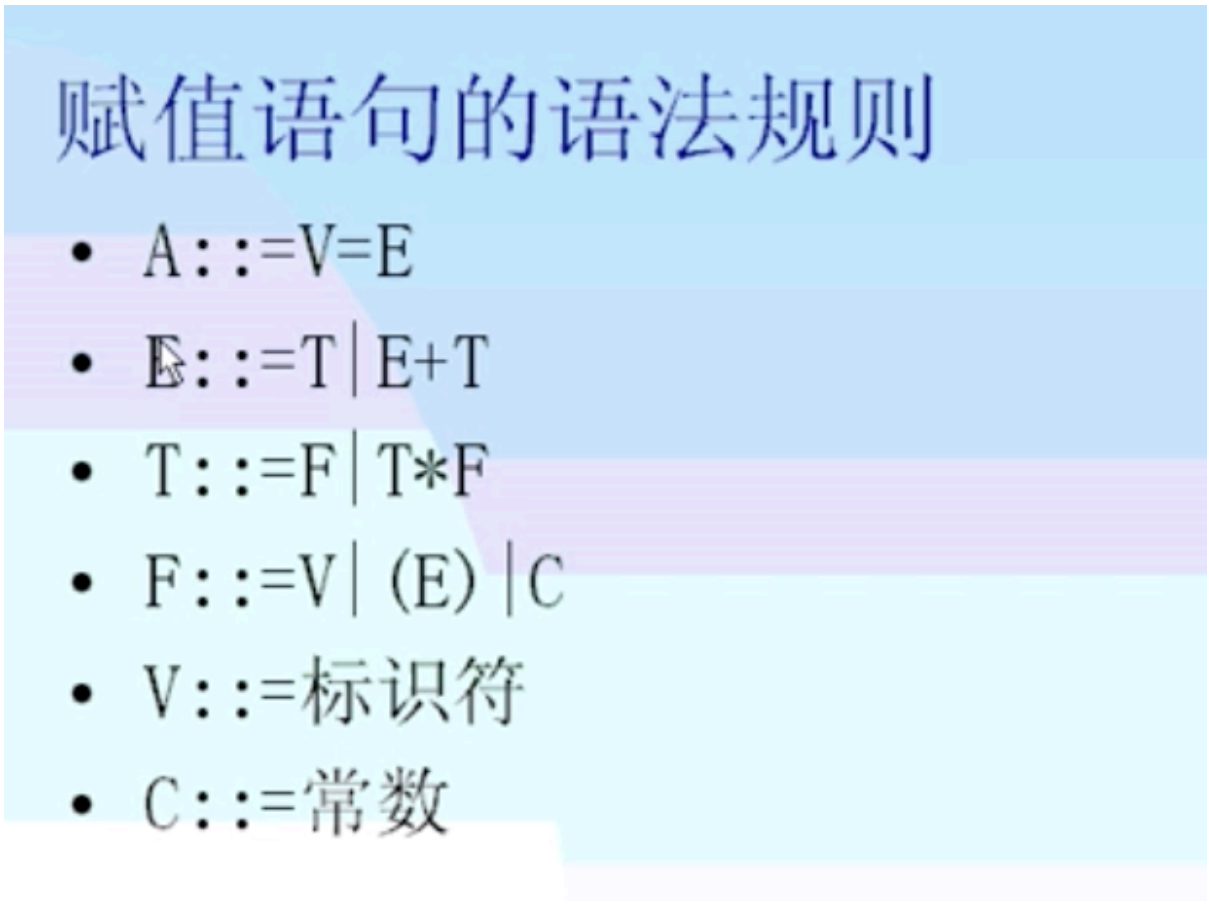
1.词法分析

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。【基本字、标识符、整常数、运算符、界限符】

- 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（转换完成后的格式：类号、内码）：
  基本字运算符界限符的转换【这种是固定的，一一对应】
  标识符的转换【(类号、内码) 类号说明是个标识符，内码表示是哪一个标识符】
  常数的转换
- 描述词法规则的有效工具是正规式和有限自动机。

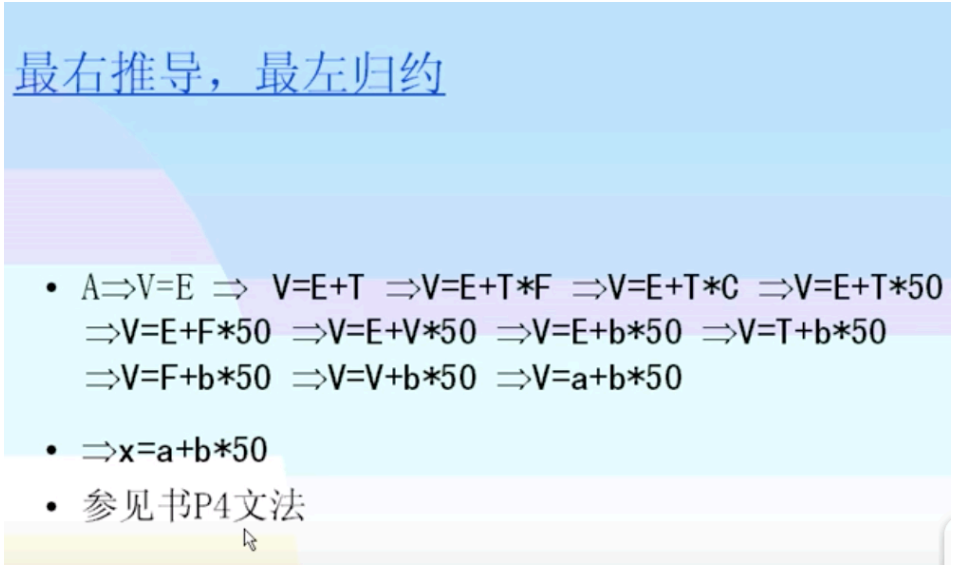
2.语法分析

- 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：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【A定义为B或C】
    <句子>::=<主><谓><宾>
    <主>::=<定><名词>

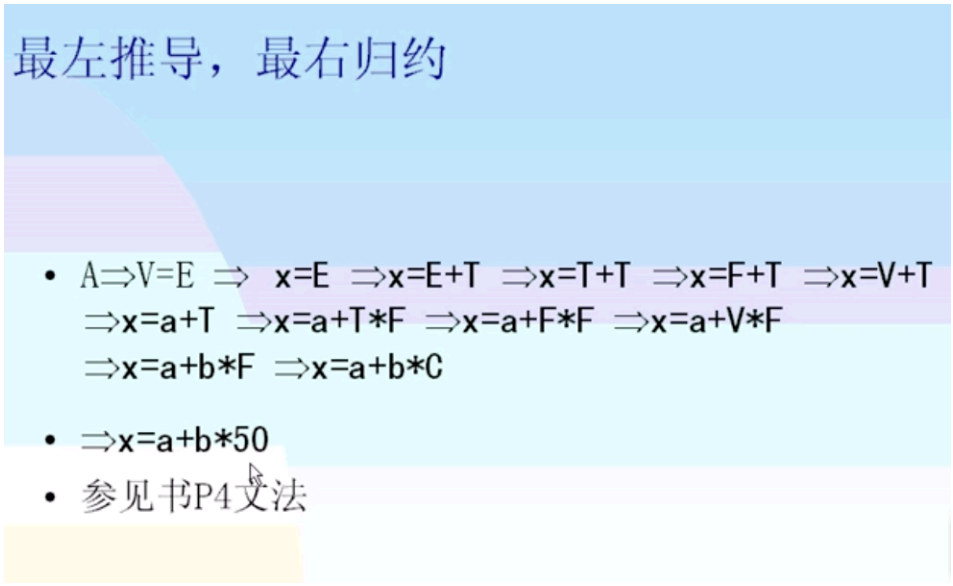


赋值号的左边一定是个标识符，不能是表达式。
赋值号右边可以是表达式(加减乘除的混合运算)
推算到底就是标识符/常数

- 语法分析的方法：
  - 推导和归约（互为逆过程）【推导和归约可以判断这个句子对不对】
    - 推导：分为最左推导和最右推导【推导是大写字母到小写字母】



- 归约：分为最左归约和最右归约【推导是小写字母到大写字母】最左就是每次都从左开始



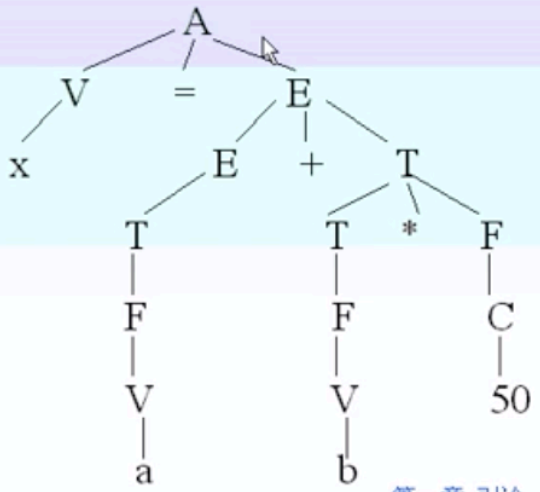
计算机可以用语法树去推。

再如：

- C语言语句 y=c+)d*(x+b
- 分析过程
- 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

【判断句子对不对就看最后推导出的语法树的叶子结点对不对】

3.语义分析和中间代码生成

中间代码就相当于翻译过程中的初稿。

- 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(看含义正不正确)
  - 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

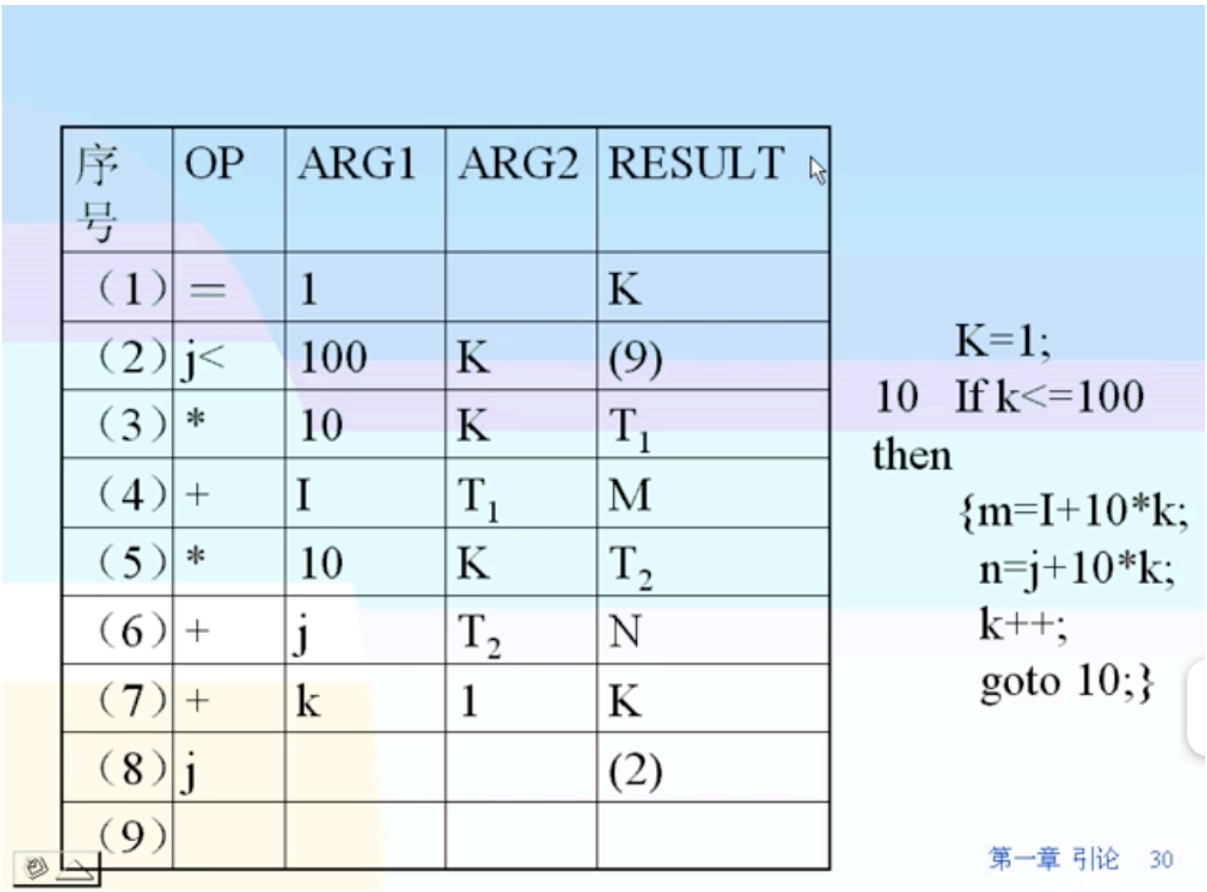
【**中间代码表** 这是个四元，算符、左操作数、右操作数、结果】

- 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

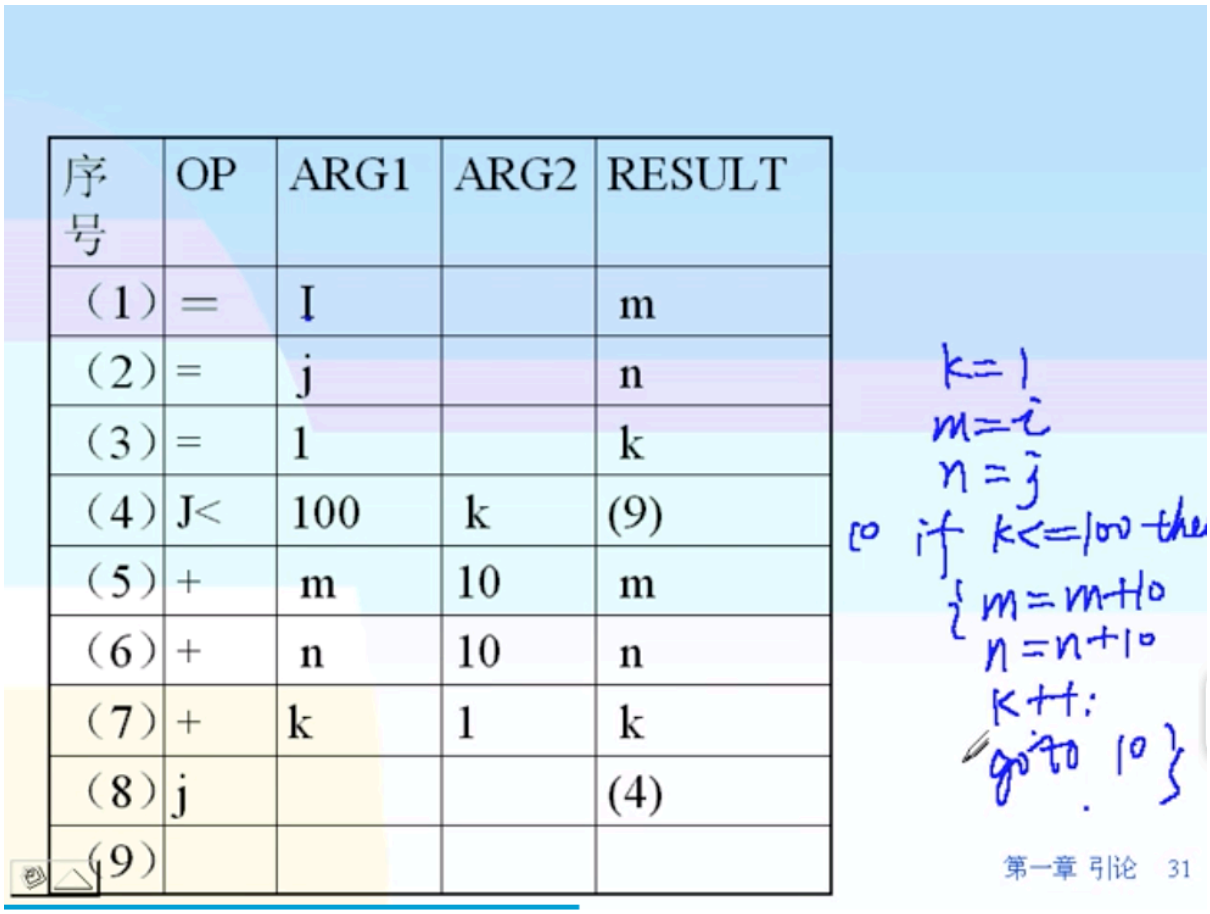
4.优化

- 任务：对前面产生的中间代码进行加工交换，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



[可以看出第3句和第5句一样，中间没对K进行处理过]

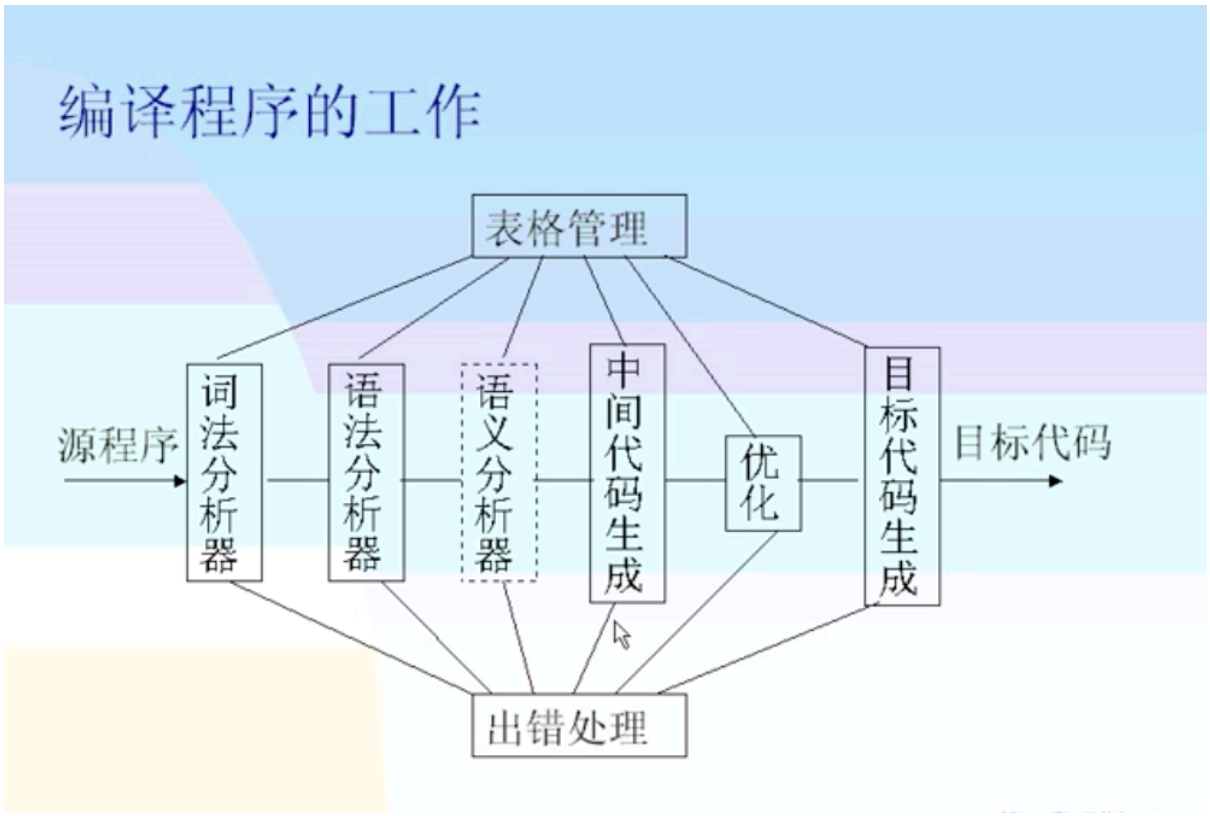
[可以看出每次m都只是增加10，可以把I放到循环外，然后m=m+10]



[可见，循环中的操作减少了，前面那种有200次加法200次乘法，后面这种只有200种加法，优化了。所以编译其实可以帮助程序员做一些优化]
[优化过的四元式再去产生目标代码，效率提高]

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。[exe代码，即01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。[只产生汇编代码是为了可针对不同机器]
    - 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，装入主存后才能称为可以运行的绝对指令代码。[需要link一些库]



[表格管理][出错处理]

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。[产生表格的只有编译过程中前3个阶段：词法分析，语法分析，中间代码生成]
- 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.
        - 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。[词法分析过程中产生]

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

- 常数表与标号表[词法分析时产生][后面一直要维护]

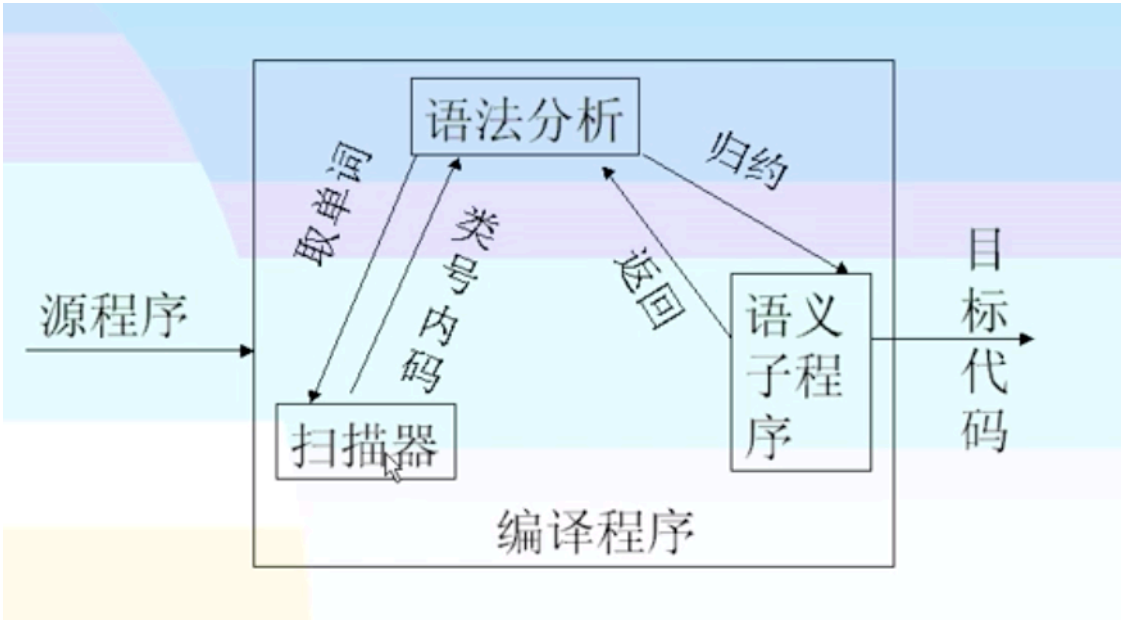
- 入口名表
  - 作用：登记过程的层号，分程序符号表入口等
- 中间代码表[往往是四元式，前面说过了]

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。[语义错误一般指不能做到的任务，比如除以0之类的]
  - 【逻辑错误无法检测】

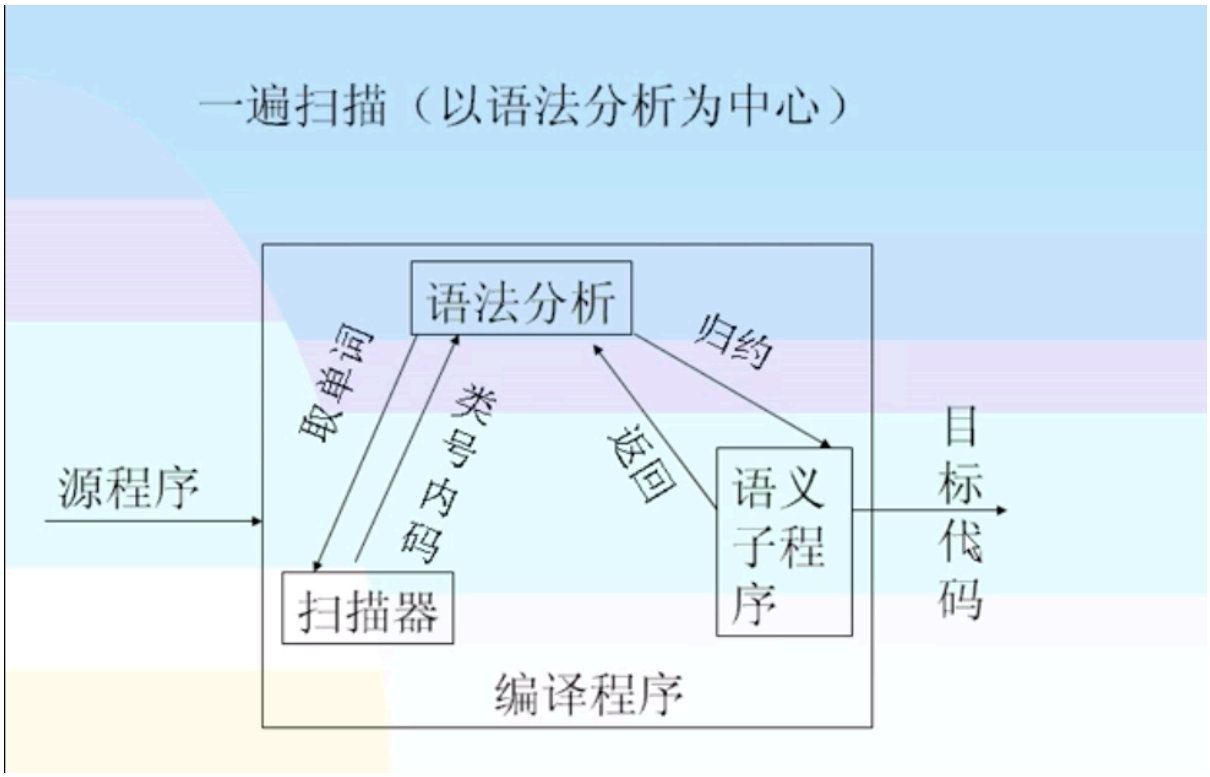
8.遍

- 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。
- 一遍扫描：



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。【与语法分析语义分析那些阶段无关系】
- 多遍扫描：
    - 优点：功能独立，结构清晰，利于优化，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。



1.3编译程序生成

1. 直接用机器语言编写的程序
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（比如c语言编写的编译程序，前提是有个c语言编译器）
   注：这是普遍采用的方法
4. 自编译【比如先做一个小核心，然后自己不断编译自己】
5. 编译工具
   --LEX(词法分析)与YACC(用于自动给产生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