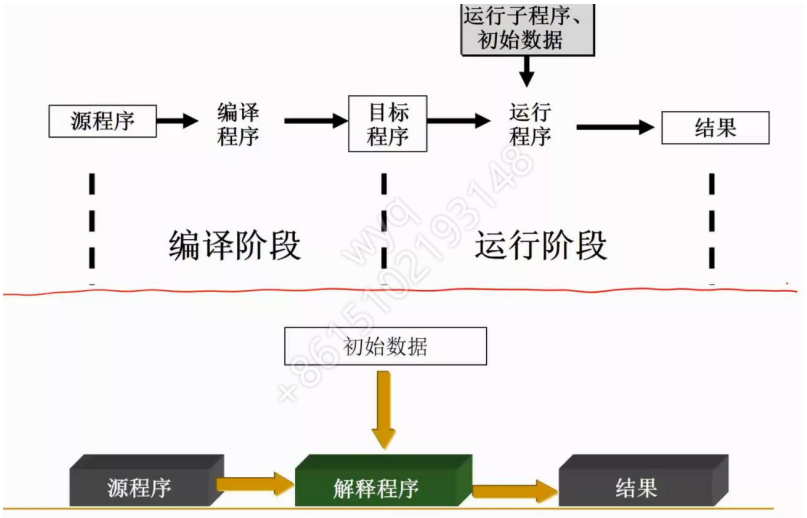
第 1 章 引论（概念 &习题）

1.1 什么是编译程序

区分 编译 和 解释

- 汇编器：汇编语言 到 机器语言的翻译。
- 高级语言 编译 低级语言 / 机器语言
- **编译程序 ： 语言翻译程序。把一种高级语言（源语言）书写的程序翻译成另一种低级语言（目标语言）的等价程序**
- 编译途径 和 解释途径（高级语言程序的处理方法）



- 解释程序
  - **不产生目标程序文件**
  - 不区别翻译阶段和执行阶段
  - 解释源程序的每条语句后直接执行
  - 程序执行期间一直有解释程序守候
  - 适合程序员以交互方式工作
- 编译程序 VS 解释程序
  - 逻辑比较

  - 空间比较

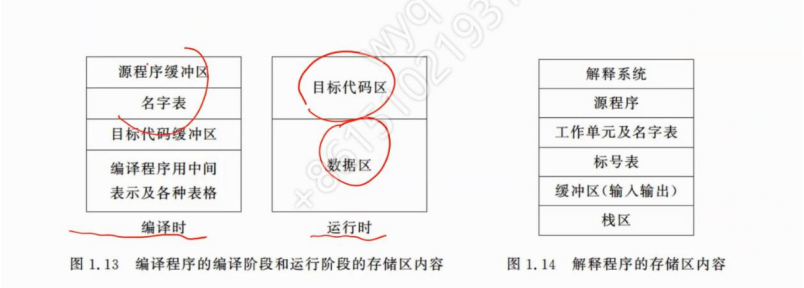
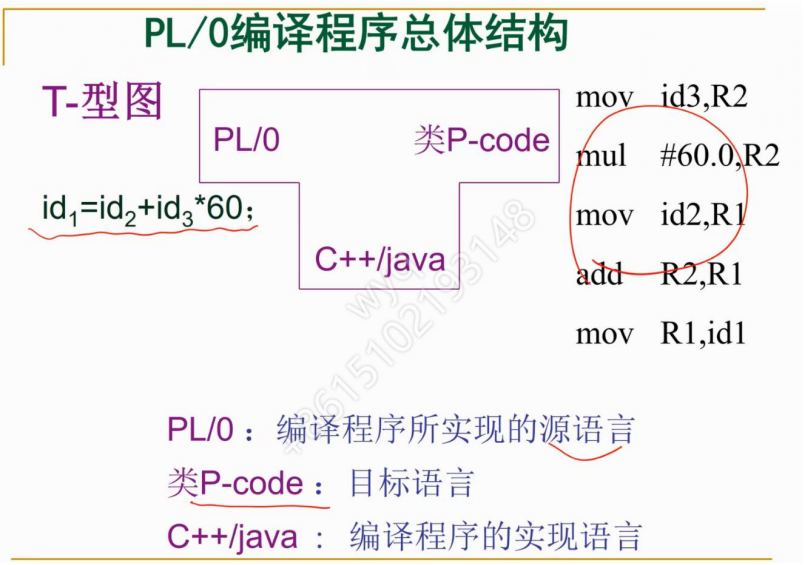


图 1.13 编译程序的编译阶段和运行阶段的存储区内容　　图 1.14 解释程序的存储区内容

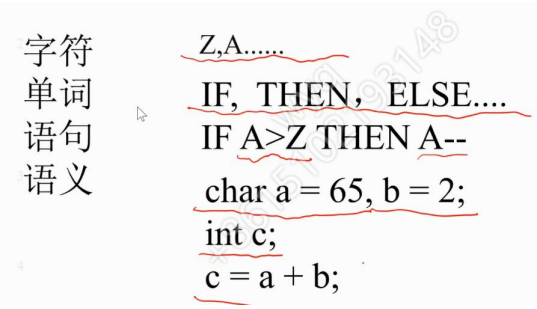
解释：运行慢，占用空间大

- PL/0 编译程序总体结构



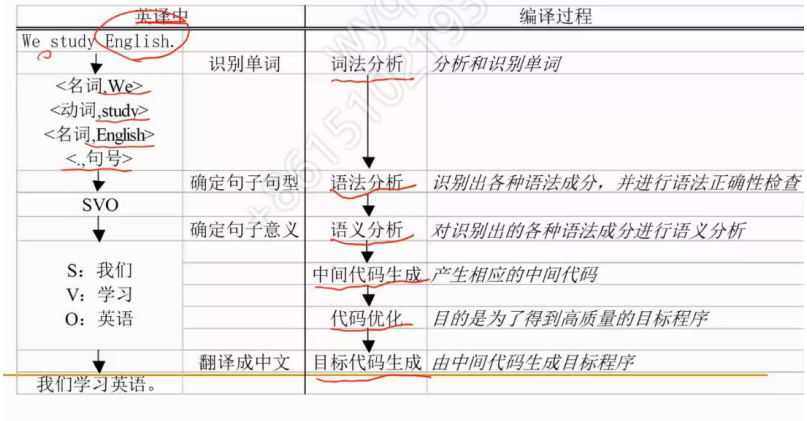
1.2 编译过程和编译程序的结构

- 1.2.0 *程序设计语言的组成(补充)



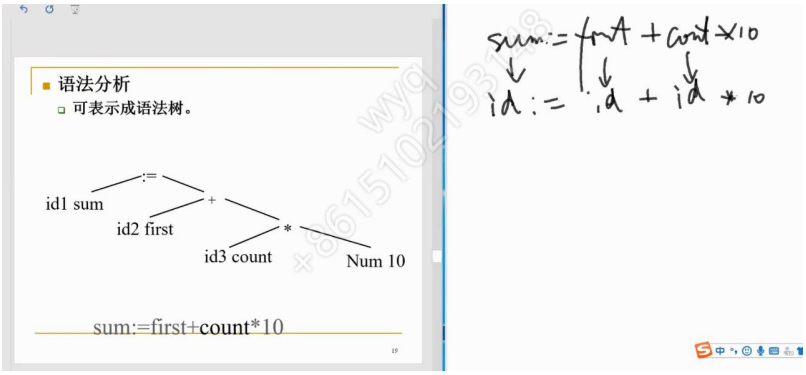
1.2.1 编译过程概述

- 图例



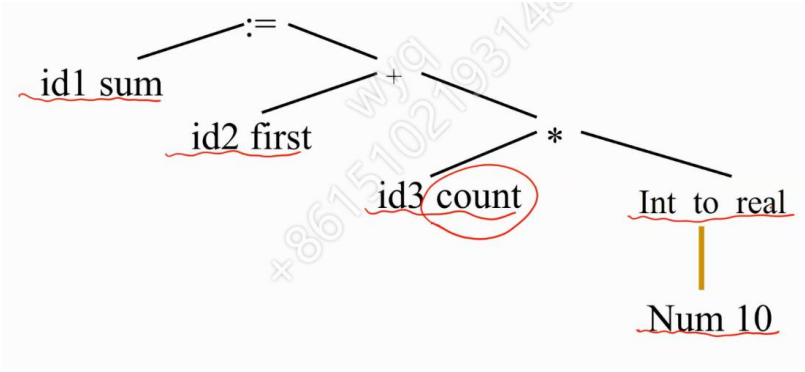
- **1. 词法分析**
  - 取单词，识别单词类型
- **2. 语法分析**
  - 单词是否能构成语句，构成什么样的语句：“语句”，“表达式”，“程序”
  - **例：i--;在 C++中是句子，在 PL0 中不是句子。**

- 语法树

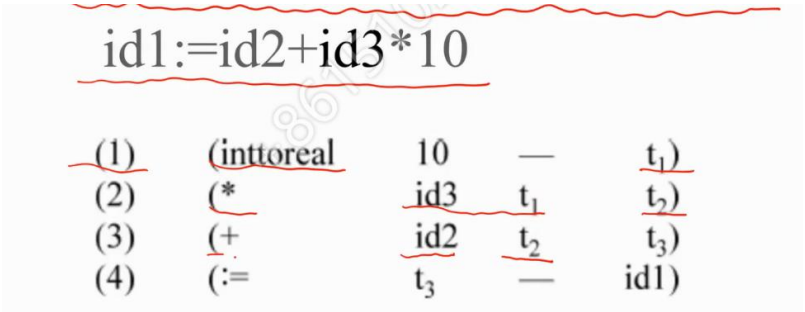


先乘除后加减；画不出语法树，则语法不正确

- **3. 语义分析**
  - 审查源程序有无语义错误，为代码生成阶段手机类型信息



  - 例：类型转换，死循环（错误），函数名写错（错误）
  - 要进行**上下文**结合，不然得不出 count 和 Num 是不一样的类型
- **4. 中间代码产生**
  - 一种结构简单、含义明确的记号系统
  - 四元式：（运算符，运算对象 1，运算对象 2，结果）



- **5. 代码优化**
  - 目的是为了得到高质量的目标程序

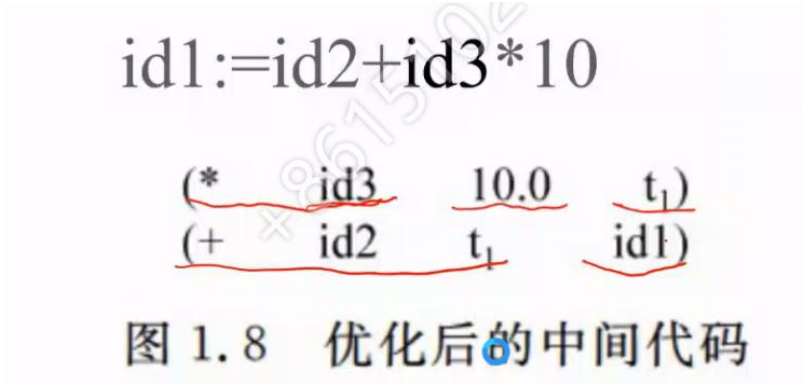
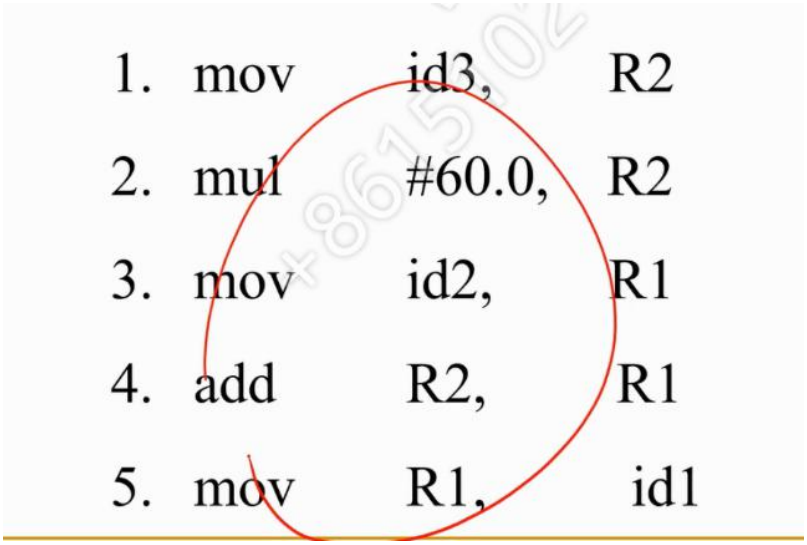


图 1.8 优化后的中间代码

- **6. 目标代码生成**
  - 由中间代码生成目标程序



- **1.2.2 编译程序的结构**

- 练习

1. 解释下列术语：
编译程序，源程序，目标程序，编译程序的前端、后端和遍
2. 编译程序有哪些主要构成成分？各自的主要功能是什么？
3. 什么是解释程序？它与编译程序的主要不同是什么？
4. 对下列错误信息，请指出可能是编译的哪个阶段（词法分析、语法分析、语义分析、代码生成）报告的。
（1）else 没有匹配的 if。
（2）数组下标越界。
（3）使用的函数没有定义。
（4）在数中出现非数字字符。
5. 通过 1.4 节的介绍以及对附录 A 中源码的初步阅读，要求读者：
（1）熟悉 PL/0 编译程序的源语言和目标语言；
（2）了解 PL/0 编译程序的基本结构；
（3）了解 PL/0 语言编译系统驱动程序的基本结构。

- 2. 词法、语法、语义、中间代码生成、代码优化、目标代码生成
- 3. 解释：一条条立即做；编译：编译 + 运行。 有中间代码生成
- **（易考填空）4. 1）语法 2）语义 3）语义 4）词法**

1.2.3 编译阶段的组合

- **前端**：和**源程序**有关的（词法分析，语法分析，语义分析，中间代码，代码优化）
- **后端**：和**目标机**有关的（目标代码生成）
- 相同前端，多后端：一个语言，不依赖某一种机器
- 多前端，相同后端：一个机器可以运行多种语言
- **一遍/多遍扫描**：编译程序按其完成规定任务的过程中对源程序或其等价的中间语言程序从头到尾扫视的次数

  一遍扫描：占用空间大，但速度快　　多遍扫描：占用空间小，独立性好，但速度慢

1.4 PL/0 编译系统

- 预处理程序
- 汇编程序
- **文法描述**

表 1.1　PL/0 语言语法的 EBNF 描述

PL/0 语法单位	EBNF 描述
<程序>	::=<分程序>.
<分程序>	::=[<常量说明部分>][<变量说明部分>][<过程说明部分>]<语句>
<常量说明部分>	::=**const**<常量定义>{,<常量定义>};
<常量定义>	::=<id>=<integer>
<变量说明部分>	::=**var**<id>{,<id>};
<过程说明部分>	::=<过程首部><分程序>{;<过程说明部分>};
<过程首部>	::=**procedure** <id>;
<语句>	::=<赋值语句>|<条件语句>|<当型循环语句>|<过程调用语句>|<读语句>|<写语句>|<复合语句>|<空语句>
<赋值语句>	::=<id>:=<表达式>
<复合语句>	::=**begin**<语句>{;<语句>} **end**
<空语句>	::=ε
<条件>	::=<表达式><关系运算符><表达式>|**odd** <表达式>
<表达式>	::=[+|−]<项>{<加减运算符><项>}
<项>	::=<因子>{<乘除运算符><因子>}
<因子>	::=<id>|<integer>|'('<表达式>')'
<加减运算符>	::=+|−
<乘除运算符>	::=*|/
<关系运算符>	::==|#|<|<=|>|>=
<条件语句>	::=**if**<条件>**then** <语句>