《1-编译器简介》学案

1. **学习目的**

能力获取：了解编译器各部分的功能，利用编译工具获得、分析各部分输出结果的能力，以及手工编写LLVM IR程序的能力。

学习线索：

首先思考问题：如何将源语言程序转换为目标语言程序？

从自然语言翻译获取灵感：“理解源程序”——“转换为目标程序”

进一步，理解源程序可以分为几步来做？转换步骤呢？

1. **学习内容**
2. 引子
   1. 对照编译过程和自然语言翻译过程，初步认识编译器的几个步骤，了解一些基本概念。
   2. 针对浏览器、数据库系统处理过程，理解“广义”和“狭义”的编译器——编译技术的广泛用途。

**重点&难点：语法树的概念及其在编译过程中的重要作用，理解其结构、有能力画出语法树。**

1. 编译器粗略构成及其广泛应用
   1. 理解“分析—综合”模型。
   2. 理解编译辅助工具都利用到了这种“分析—综合”的编译技术。
   3. 理解这种模型还可用到其他很多应用中。

**重点&难点：相关软件如何用“分析—综合”模型实现其功能。**

1. 分析阶段
   1. 词法分析：“单词”（token）的概念。
   2. 语法分析：语法树的概念，“语法范畴”的概念引出文法概念。
   3. 语义分析：类型检查、类型转换。
   4. Latex中的分析。

**重点&难点：几个阶段的功能及输入输出。理清基本概念，单词是一个类别概念，词法分析的输出和语法分析过程中都只关注类别（NUM）、而不关注实例个体（12、3.4、…）；语法树是程序中语法结构层次关系的表达，单词（和其他语法范畴）抽象出更上层的语法范畴，通常，这种抽象或者说定义包含递归。**

1. 完整六个阶段
   1. 完整编译过程和两个贯穿始终的工作。
   2. 中间代码生成：一个桥梁，方便代码优化，了解LLVM IR。
   3. 代码优化：基本原则。
   4. 代码生成：汇编程序。

**重点&难点：思考如何“简单”有“有效”地优化代码。**

1. 语言处理系统
   1. 预处理：宏的概念和处理方式，Latex中的宏，头文件处理。
   2. 汇编器：两次扫描的概念。
   3. Loader和linker：可重定位机器码的概念。
   4. 前端和后端：软件工程角度的划分。
   5. 编译工具：后续学习内容。

**重点&难点：预处理的工作机制，会产生什么结果；为什么要两次扫描，与可重定位机器码的关系，理解程序编译、执行与操作系统紧密相关。**

1. 编译技术发展历史
2. **预习作业**
3. 动手编译P25、26程序片段，观察你的编译器给出的错误/警告。

*提示：可能需要调整编译选项来控制编译器警告级别。鼓励大家尝试静态检查工具lint（Windows版本名为splint）。*