编译原理与设计：编译器简介

北京理工大学 计算机学院

内容

- 编译器定义
- 编译器表示
- 编译器典型结构
- 遍的概念
- 常见编译器结构
- 编译器的构造

编译器定义

将某一种**程序**设计语言写的程序翻译成**等价**的另一种语言的程序的**程序**，称之为编译程序。

A **program** that accepts as input a **program** text in a certain language and produces as output a **program** text in another language, while preserving the meaning of that text (Grune *et al*, 2000)

A **program** that reads a **program** written in one language (*source language*) and translates it into an equivalent **program** in another language (target language) (Aho *et al*)

编译器定义

- 源语言： (source language)
  - 编译器输入程序的描述语言，一般是高级程序设计语言。
- 源程序： (source program)
  - 编译程序的输入程序称为源程序。
- 目标语言： (target language)
  - 目标程序的描述语言称为目标语言。
- 目标程序： (target program)
  - 源程序经过编译后生成的程序称之为目标程序。

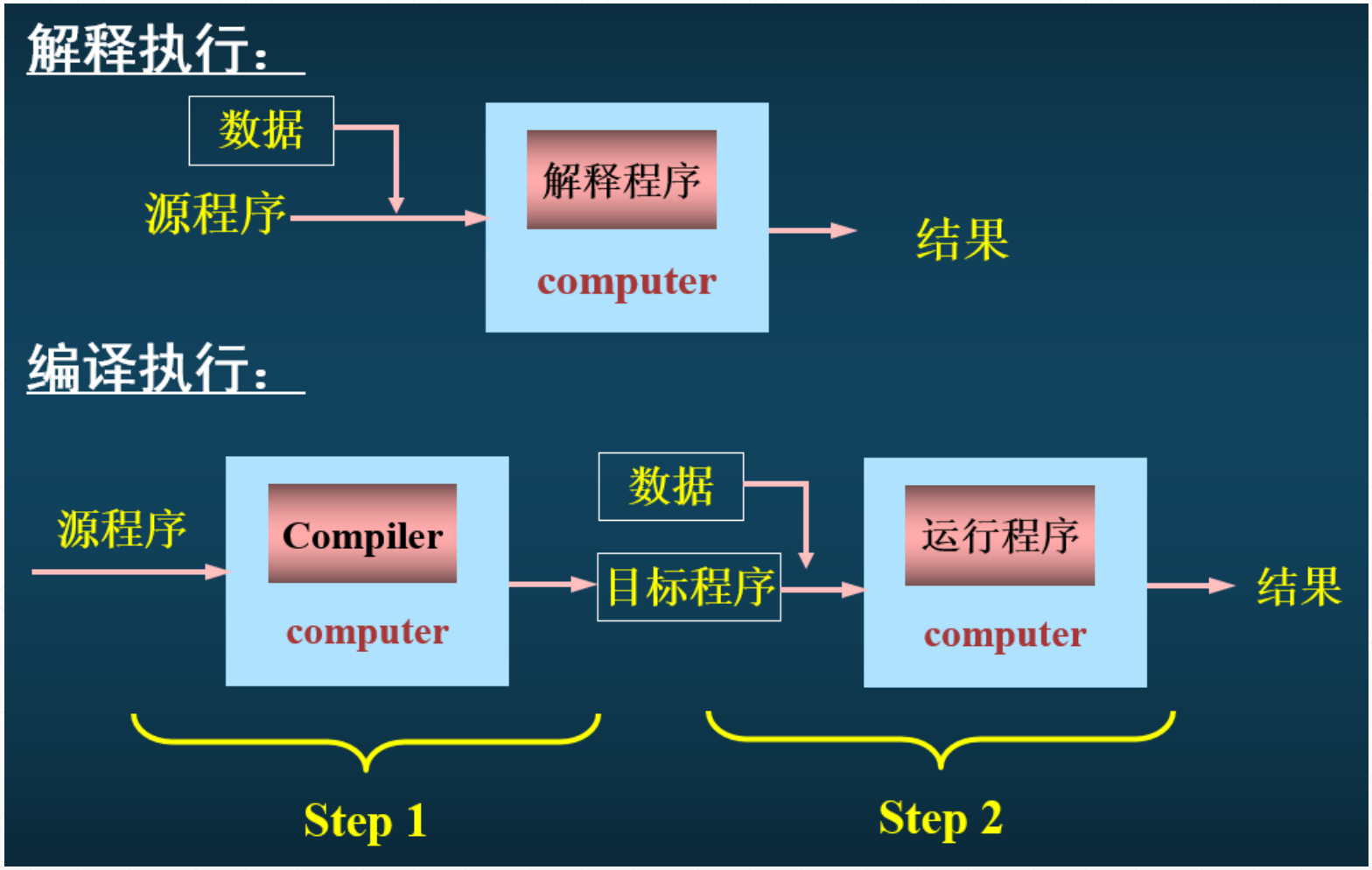
编译器定义

- 宿主语言
  - 编译程序的实现语言。
- 宿主机（目标机）
  - 编译程序的运行环境。

编译器定义

- 人、机交互的基本界面；
- 软件学科的重要分支，系统软件的重要组成部分；
- 用户直接关心的工具。

编译器定义

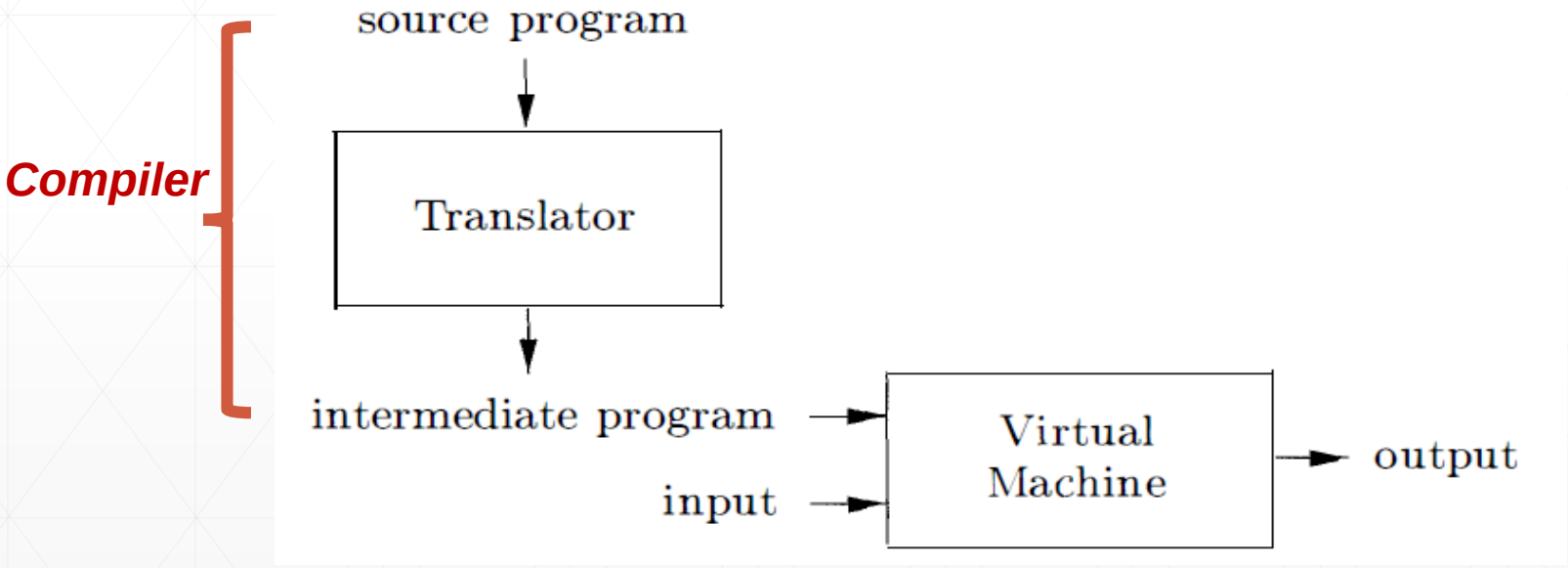

解释执行：
数据
源程序
解释程序
computer
结果
编译执行：
源程序
Compiler
computer
数据
目标程序
运行程序
computer
结果
Step 1
Step 2

编译器定义

- 常见语言
  - C is typically compiled
  - Lisp is typically interpreted
  - Java is compiled to bytecodes, which are then interpreted
- 其他编译器
  - C++ to SPARC/Pentium/... Assembly
  - C++ to C
  - High Performance Fortran (HPF) to Fortran
  - Fortran to C

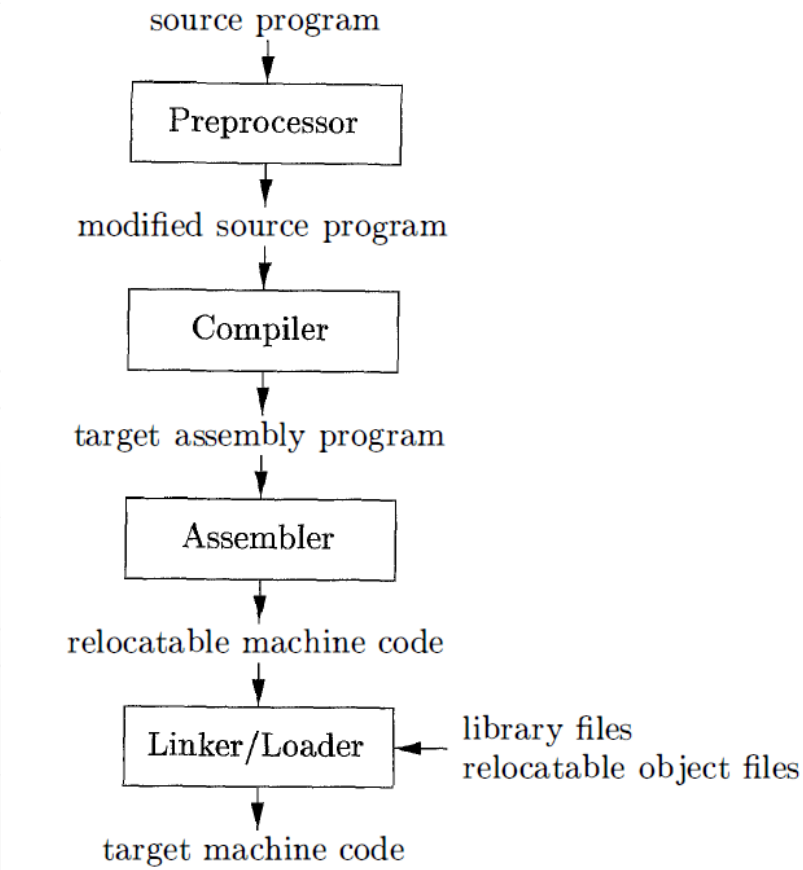
编译器定义

- Java is compiled to bytecodes, which are then interpreted



编译器定义

- C语言


source program
Preprocessor
modified source program
Compiler
target assembly program
Assembler
relocatable machine code
Linker/Loader
library files
relocatable object files
target machine code

编译器定义

编译执行是由编译程序生成一个与源程序等价的目标程序，它可以完全取代源程序，该目标程序可以运行任意次。

解释执行不生成目标程序，仅是对源程序逐句解释逐句执行。

编译执行类似于自然语言的笔译；解释执行类似自然语言的口译。

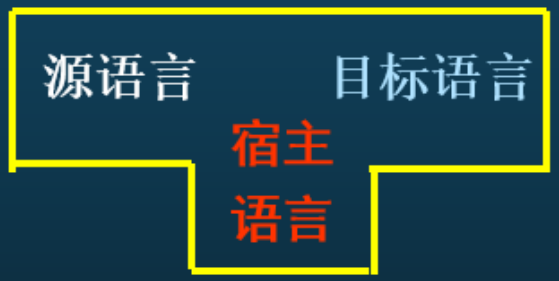
编译器表示

1. 函数表示 ：

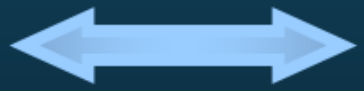
$$T = C(S)$$

T: 目标程序　　C: 编译程序　　S: 源程序

2. T型图表示 (体现编译程序的三个要素)



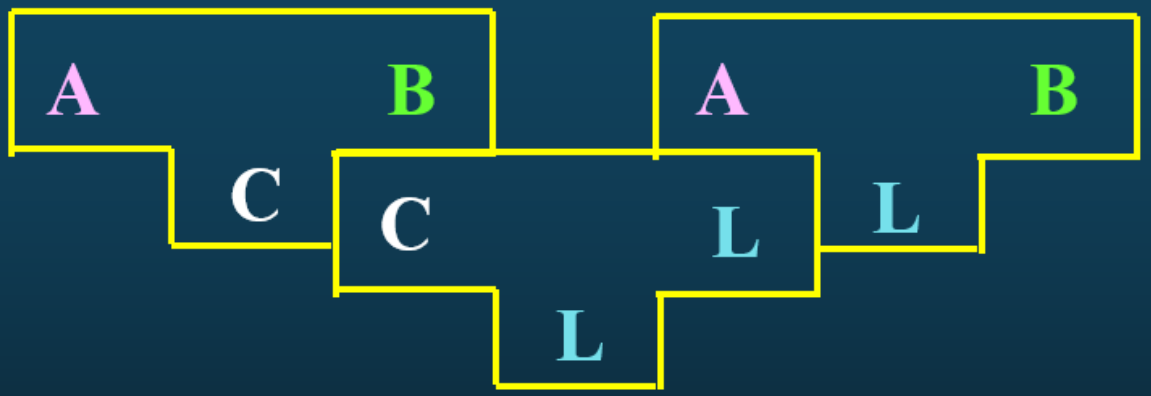
3. 符号表示



编译器表示

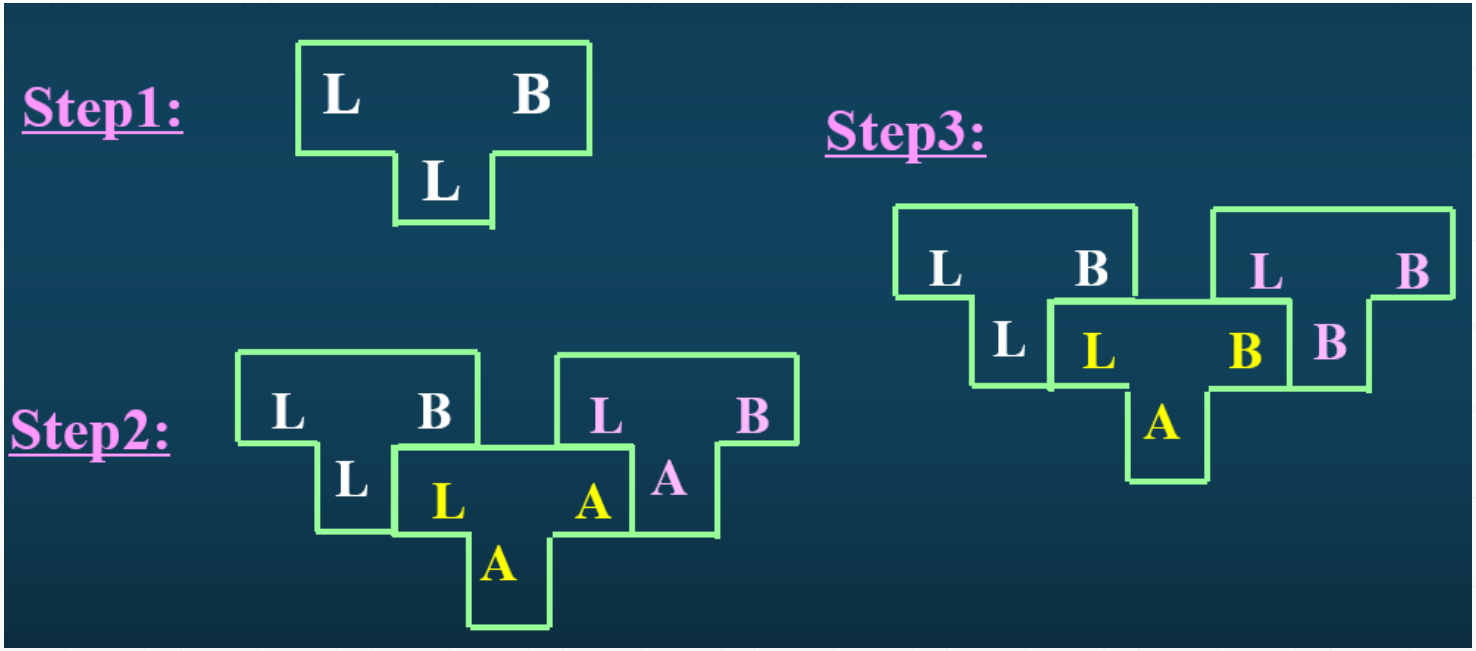
T型图的左上角表示源语言，右上角表示目标语言，而其底部表示实现语言。

- T型图联立表示



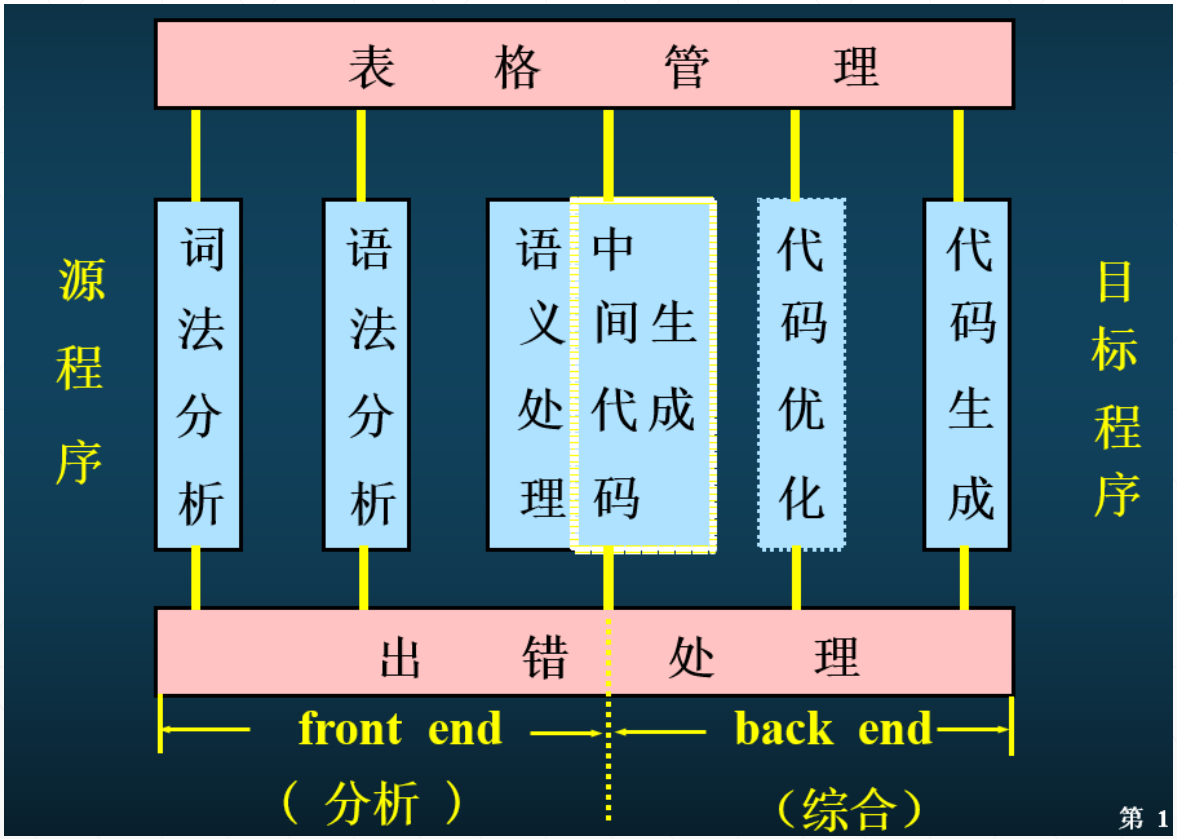
编译器表示

- 例：将A机器上已经存在的L语言的编译程序移植到B机器上。



编译器的组织与结构

- 编译程序的逻辑结构(经典划分)



编译器的组织与结构

- 词法分析 (lexical analysis)

词法分析阶段的任务是对输入的符号串形式的源程序进行最初的加工处理。 它依次**扫描**读入的源程序中的每个字符，**识别**出源程序中有独立意义的源语言单词，用某种特定的数据结构对它的属性予以表示和标注。词法分析实际上是一种线性分析。属性字的数据结构可据不同语言及编译程序实现方案来设计，但一般设计成单词属性标识及单词内码两个数据项。

编译器的组织与结构

- 词法分析

```
a[index] = 12 * 3 ;
```

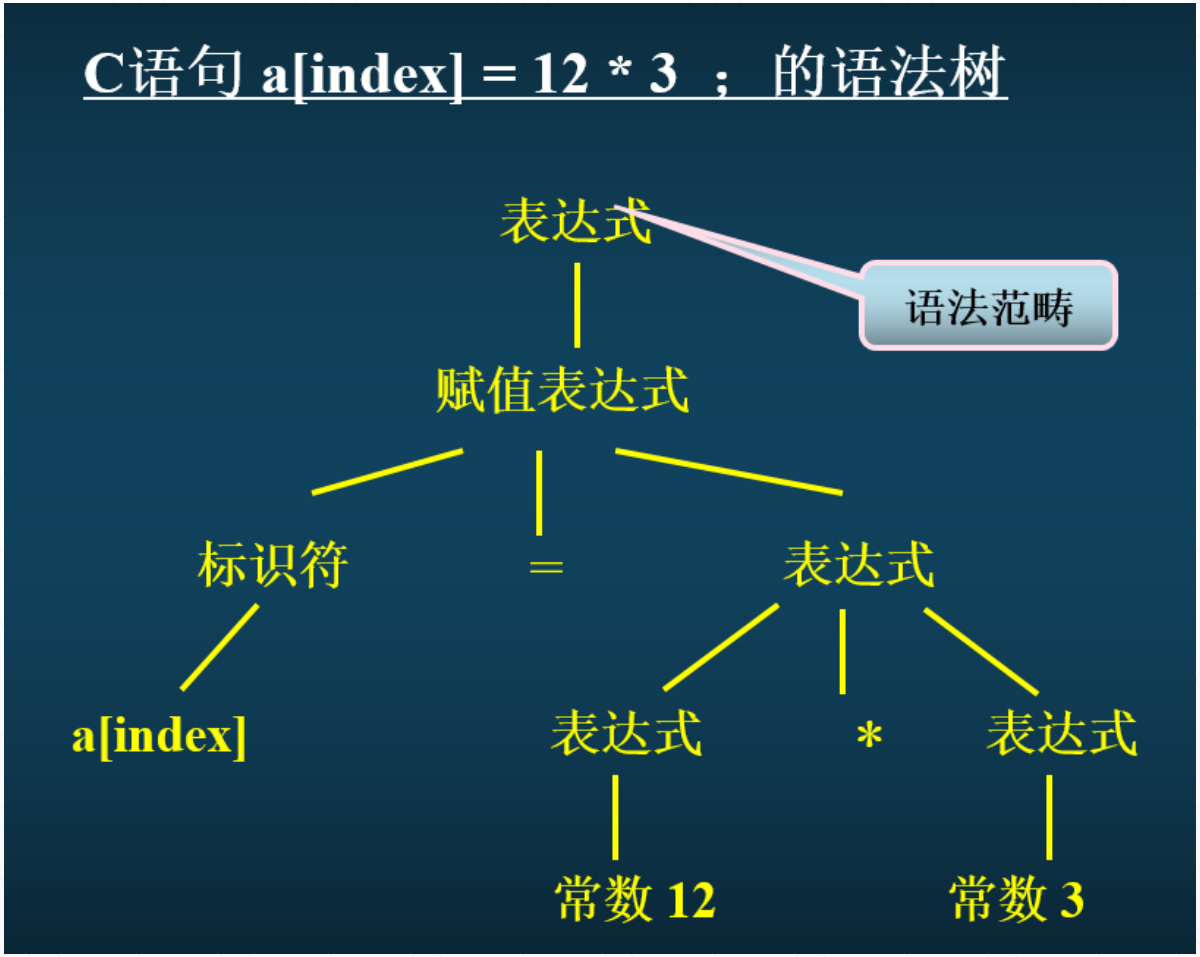
(1)	a	标识符
(2)	[	左方括号
(3)	index	标识符
(4)	]	右方括号
(5)	=	赋值
(6)	12	整常数
(7)	*	乘号
(8)	3	整常数
(9)	;	分号

编译器的组织与结构

- 语法分析 (Syntax analysis)

语法分析的任务是依据语言文本所规定的语法规则，对词法分析的结果进行语法检查，并识别出单词序列所对应的语法范畴 。通常将语法分析的过程结果表示为分析树(parse tree)或语法树(syntax tree)，它是一种层次结构的形式。

编译器的组织与结构


C语句 a[index] = 12 * 3 ； 的语法树
表达式
语法范畴
赋值表达式
标识符
=
表达式
a[index]
表达式
*
表达式
常数 12
常数 3

编译器的组织与结构

- 语义分析 与中间代码生成

语义分析阶段的任务是依据语言文本限定的语义规则对语法分析所识别的语法范畴进行语义检查和处理, 直至最后翻译成与其等价的某种中间代码或目标代码。语义分析是整个编译程序完成的最实质性的翻译任务。

编译器的组织与结构

- 语义分析 与中间代码生成

x=(a+b)*c

+	a	b	T_1
*	T_1	c	T_2
=	T_2		x

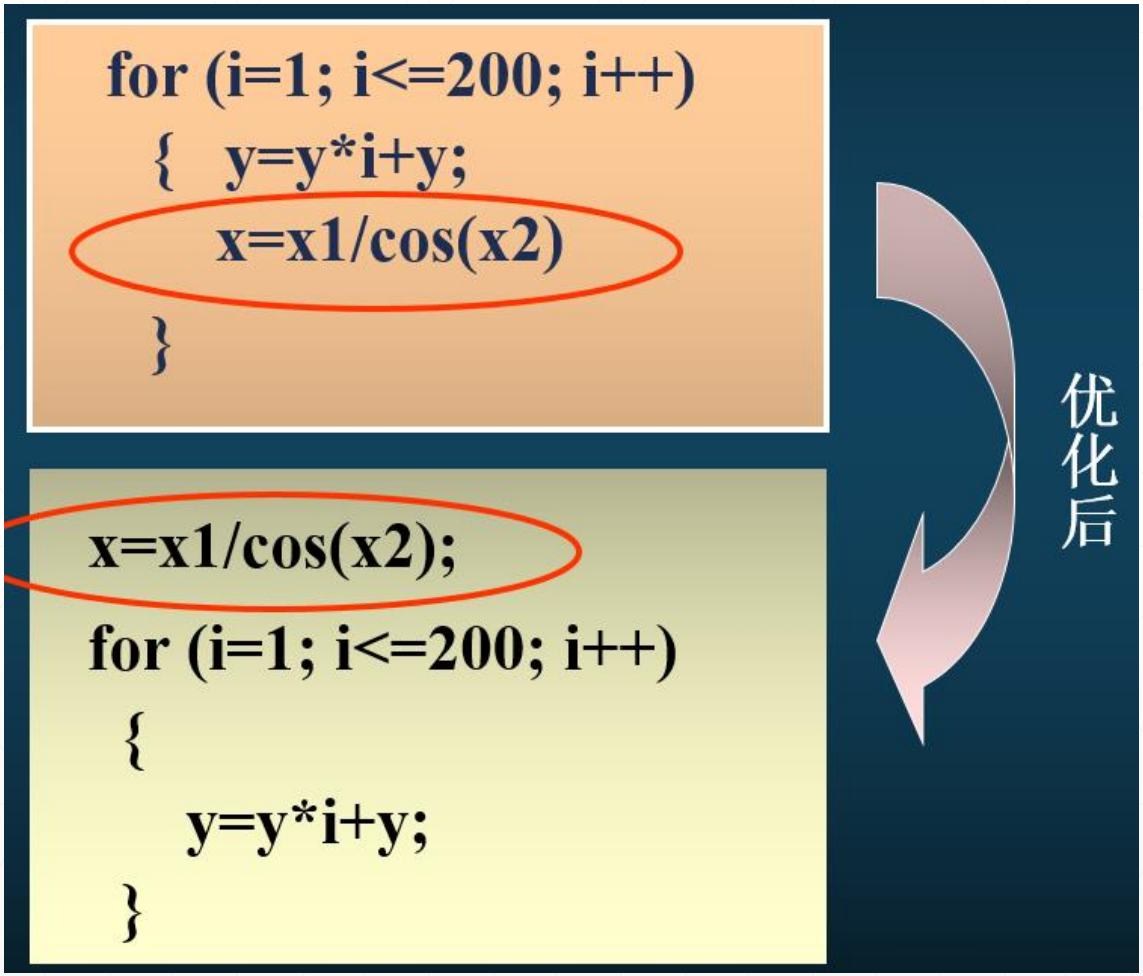
编译器的组织与结构

- 代码优化

代码优化是为了改进目标代码的质量而在编译过程中进行的工作。代码优化可以在中间代码或目标代码级上进行，其实质是在不改变源程序语义的基础上对其进行加工变换，以期获得更高效的目标代码。而高效一般是对所产生的目标程序在运行时间的缩短和存贮空间的节省而言。

编译器的组织与结构

- 代码优化



编译器的组织与结构

- 目标代码生成

根据中间代码及编译过程中产生的各种表格的有关信息，最终生成所期望的目标代码程序。一般为特定机器的机器语言代码或汇编语言代码，需要充分考虑计算机硬件和软件所提供的资源，以生成较高质量的目标程序。

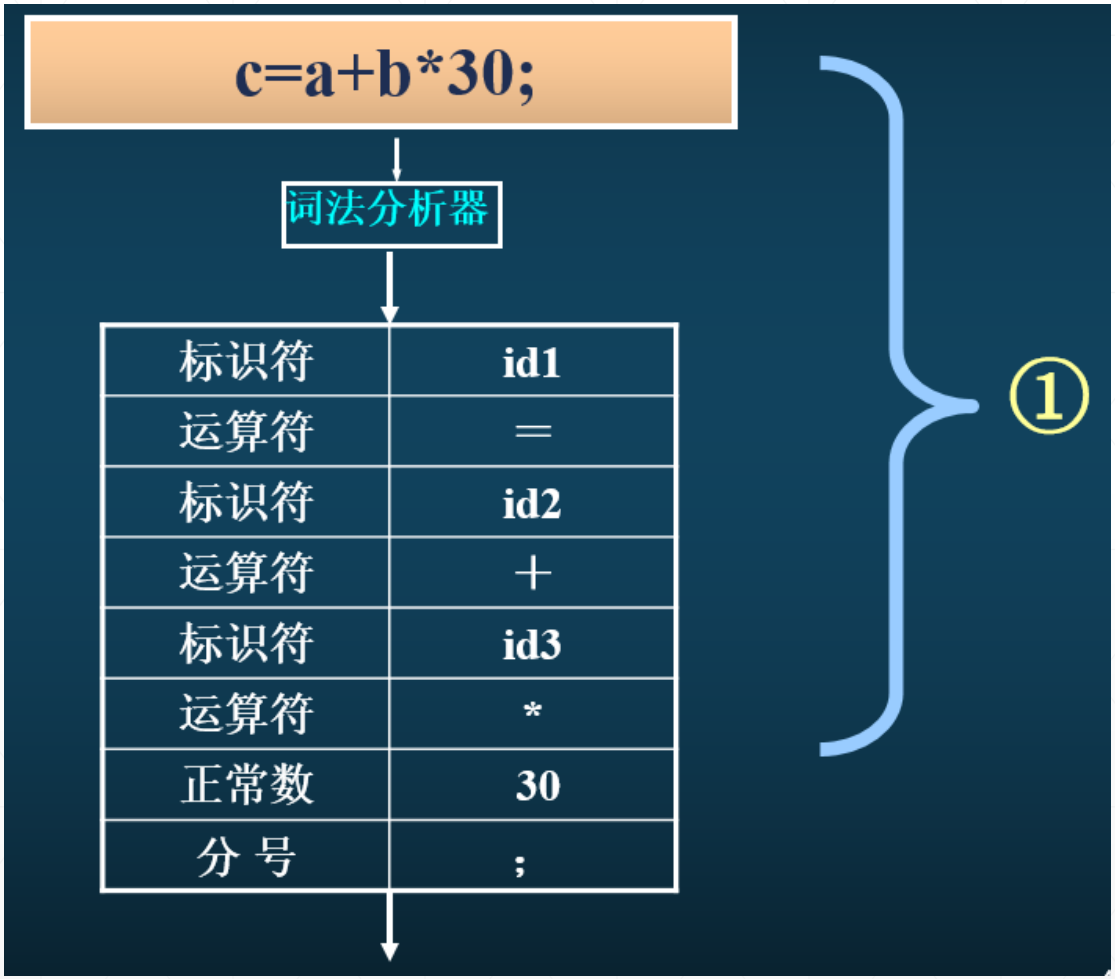
编译器的组织与结构

- 目标代码生成

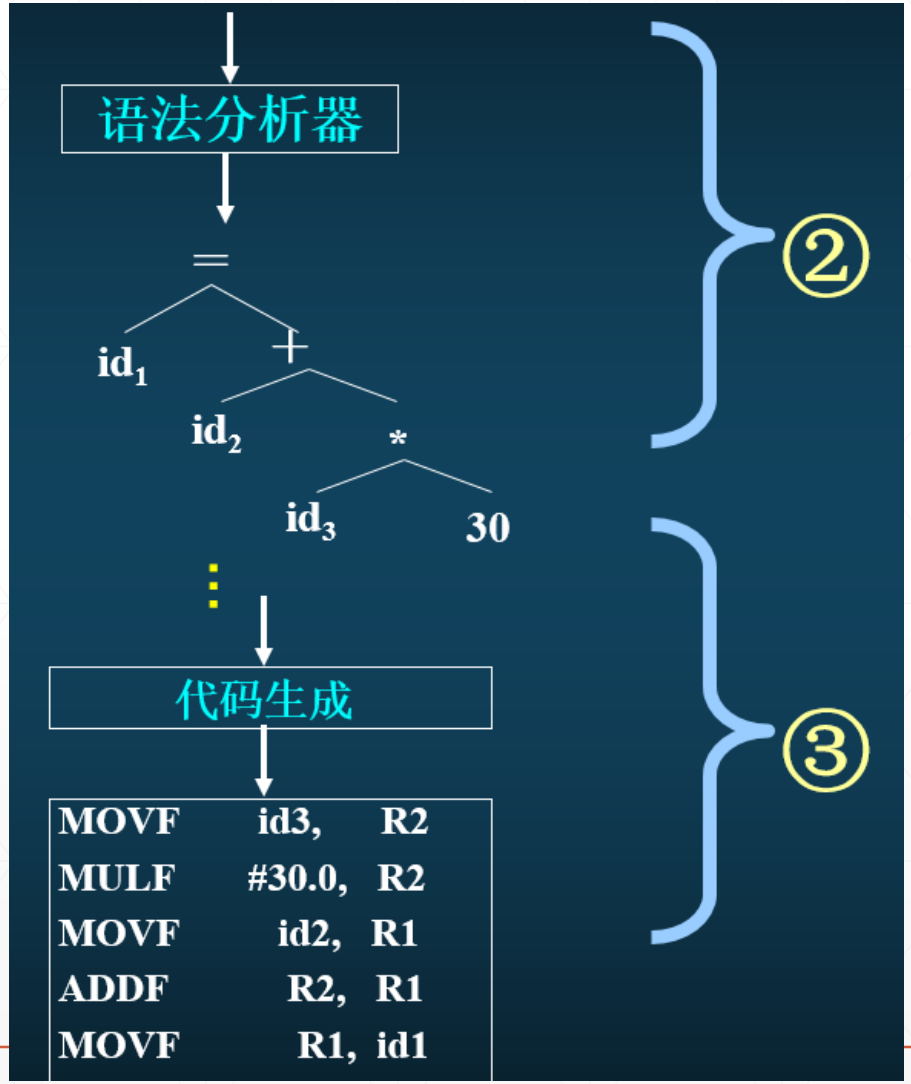
```
a[index] = 12 * 3 ;
```

```
MOV    R0,  index          ;; 索引值赋给寄存器R0
MUL    R0,  4              ;; 存储按字节编址，整型数
                           ;; 占4个字节
MOV    R1,  &a             ;; &a表示数组a的基地址
ADD    R1,  R0             ;; 计算a[index]的地址
MOV   [R1],  36            ;; a[index] = 36
```

编译器的组织与结构

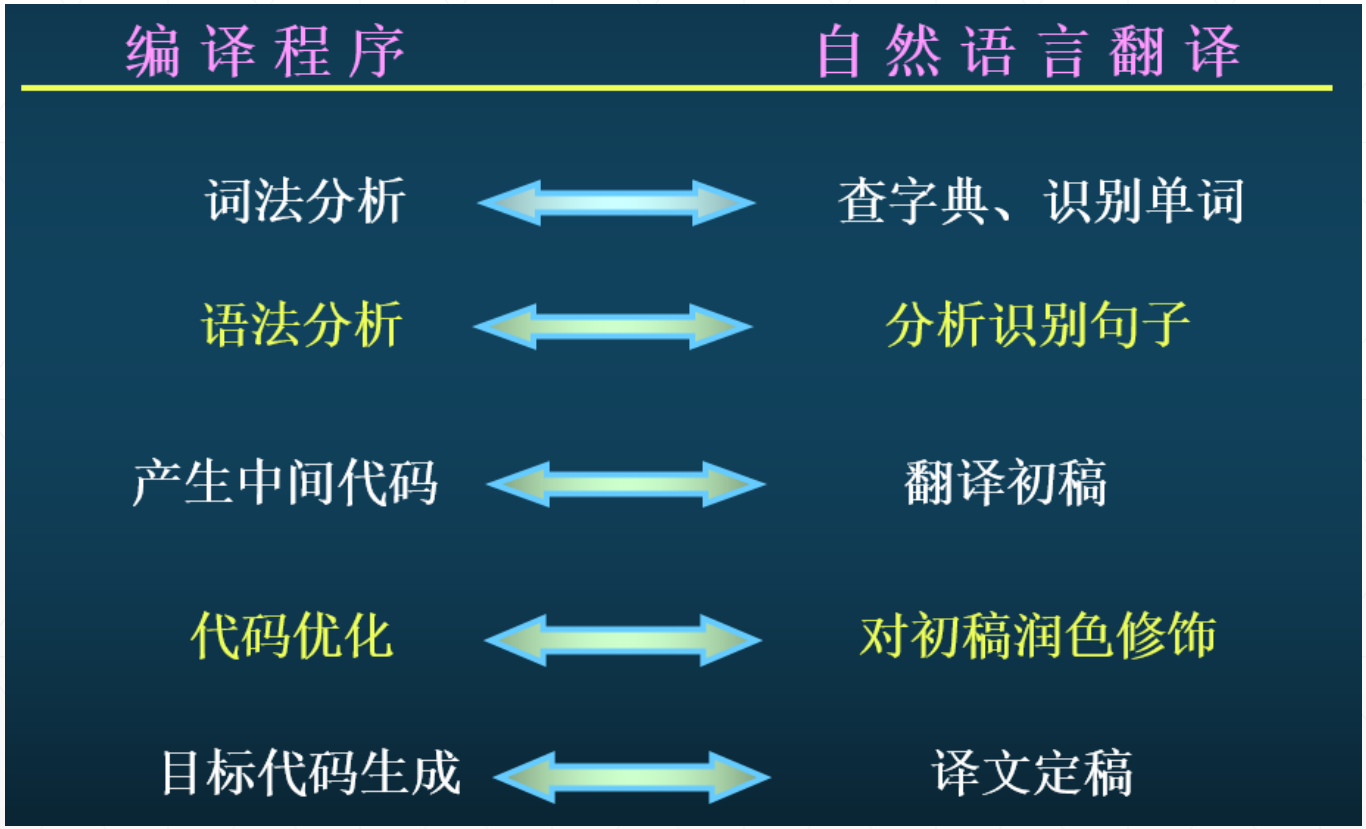


编译器的组织与结构


语法分析器
=
id1
+
id2
*
id3
30
②
③
代码生成
MOVF id3, R2
MULF #30.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1

编译器的组织与结构

- 编译阶段与自然语言的对比



编译器的组织与结构

- **表格管理**

编译程序的公共辅助程序。对源程序中的各种量进行管理，登记相应的表格。编译程序处理时通过查表得到所需的信息。

- **出错处理**

编译程序的公共辅助程序。对源程序中的各种错误检查、定位、定性、报告，并尽可能将错误限制在尽可能小的范围内，保障编译继续。

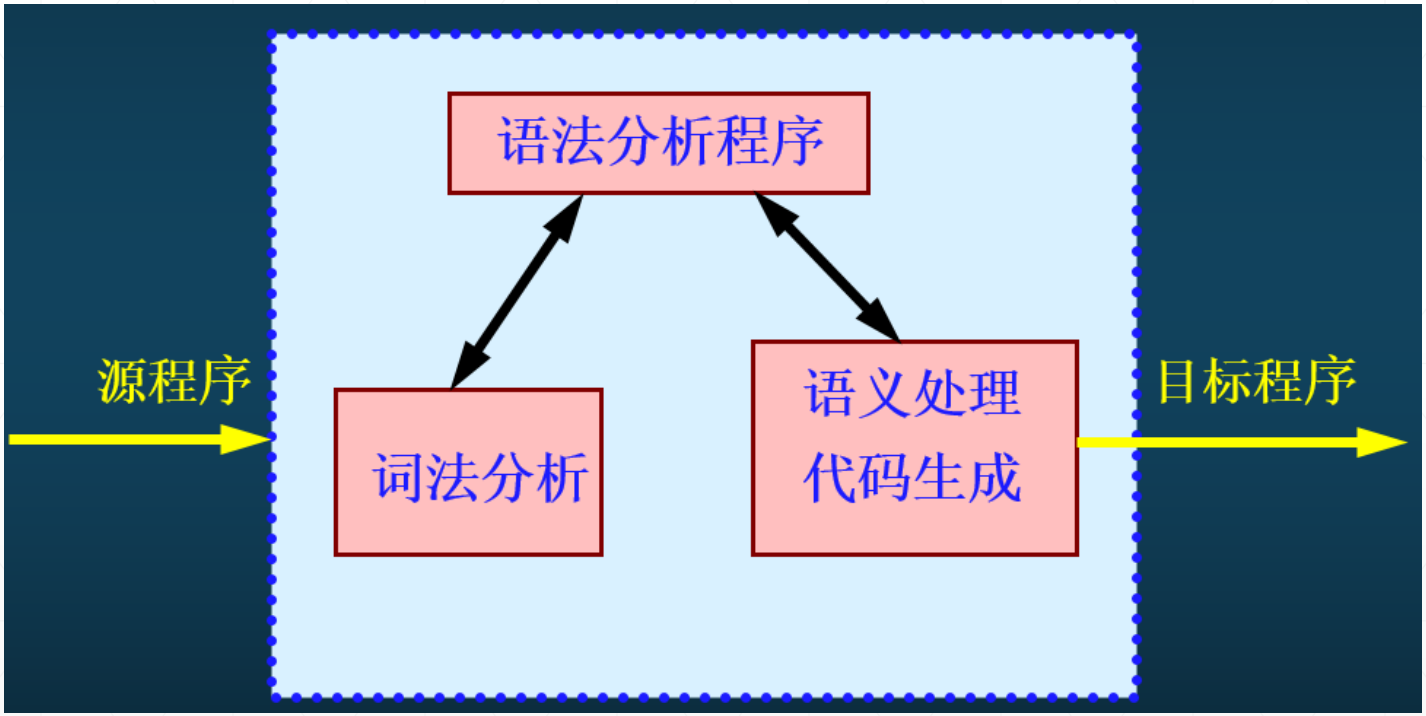
遍的概念

对源程序或源程序的中间形式从头到尾扫描一遍，并做有关的分析加工，生成新的源程序的中间形式或生成目标程序。

- 遍设置的考虑因素
  - (1) 宿主机存储容量；
  - (2) 编译程序功能的强弱；
  - (3) 源语言的繁简及约束；
  - (4) 优化因素；
  - (5) 设计、实现的环境、工具及人员因素等。

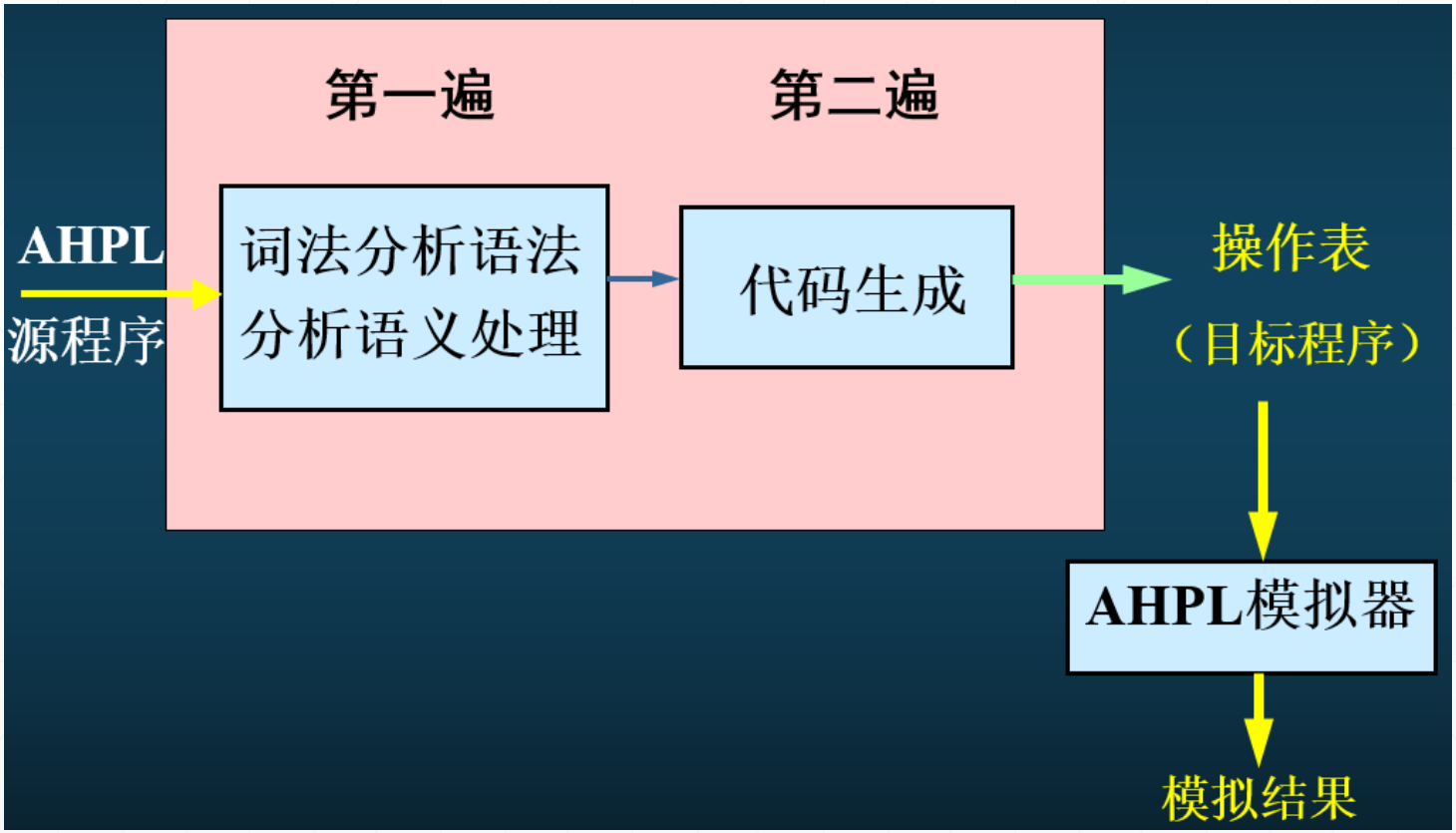
典型编译器结构

- 一遍扫描的编译程序结构模型



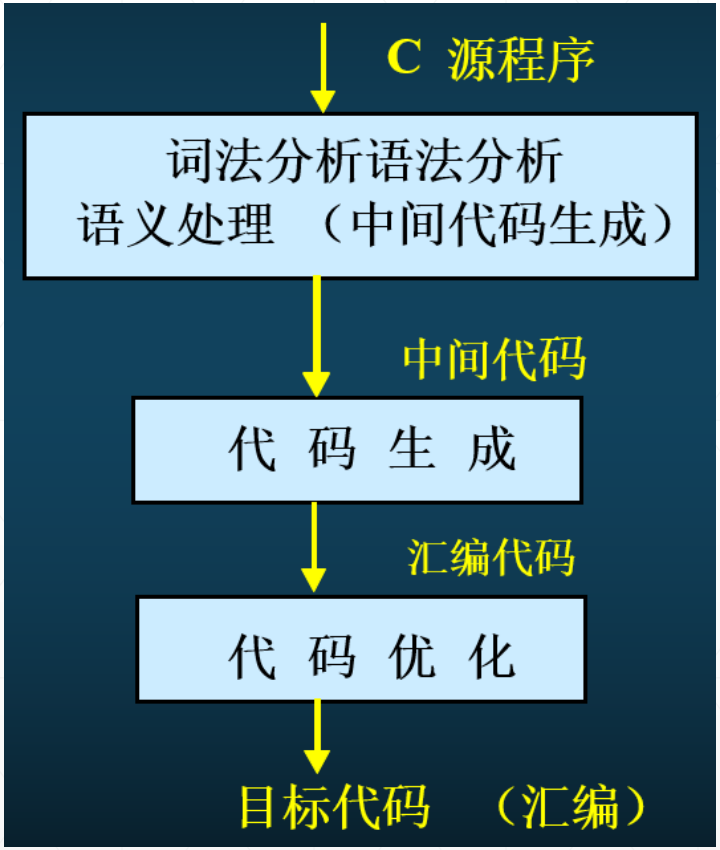
典型编译器结构

- AHPL模拟器（两遍扫描）



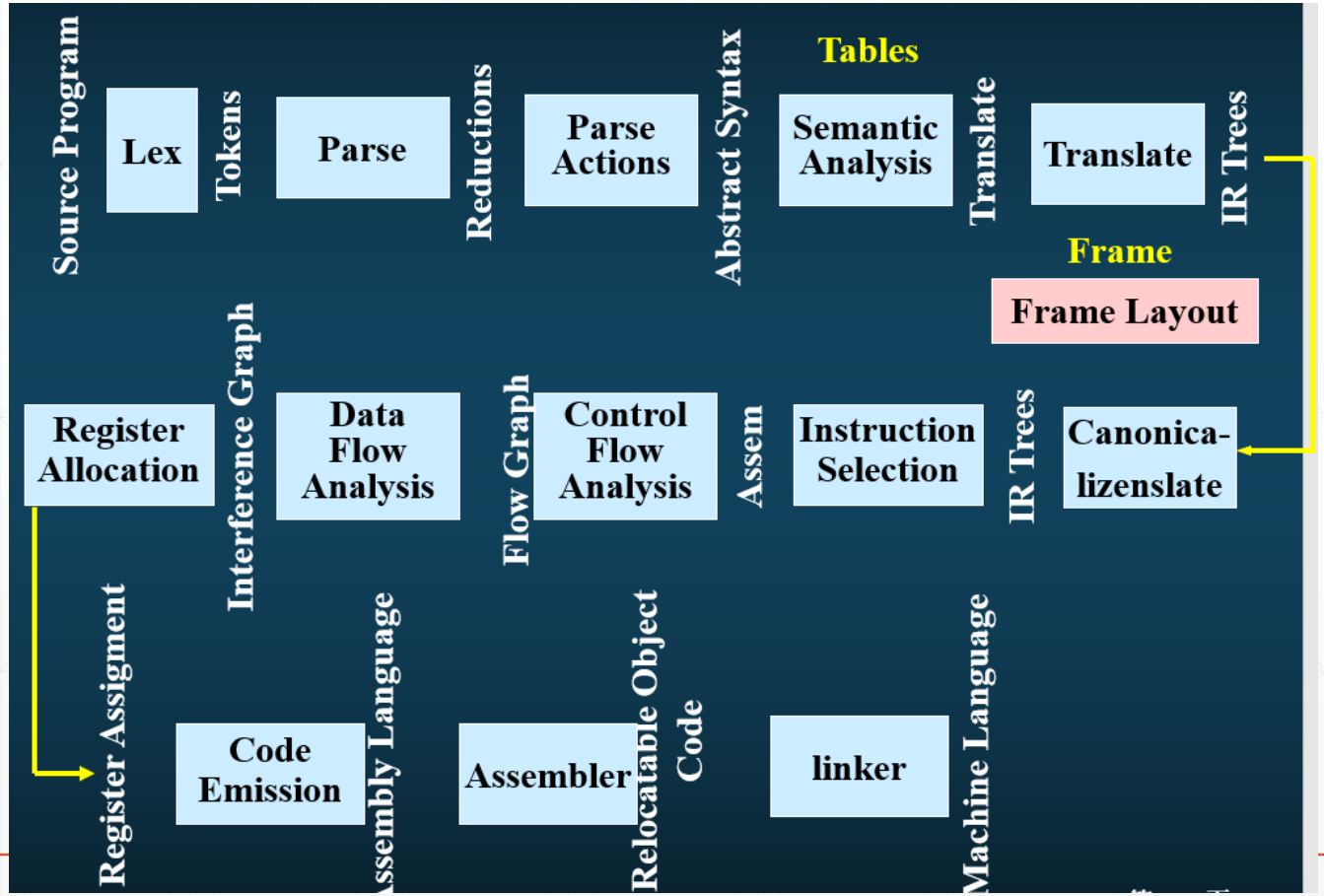
典型编译器结构

- PDP – 11 C编译器结构模型



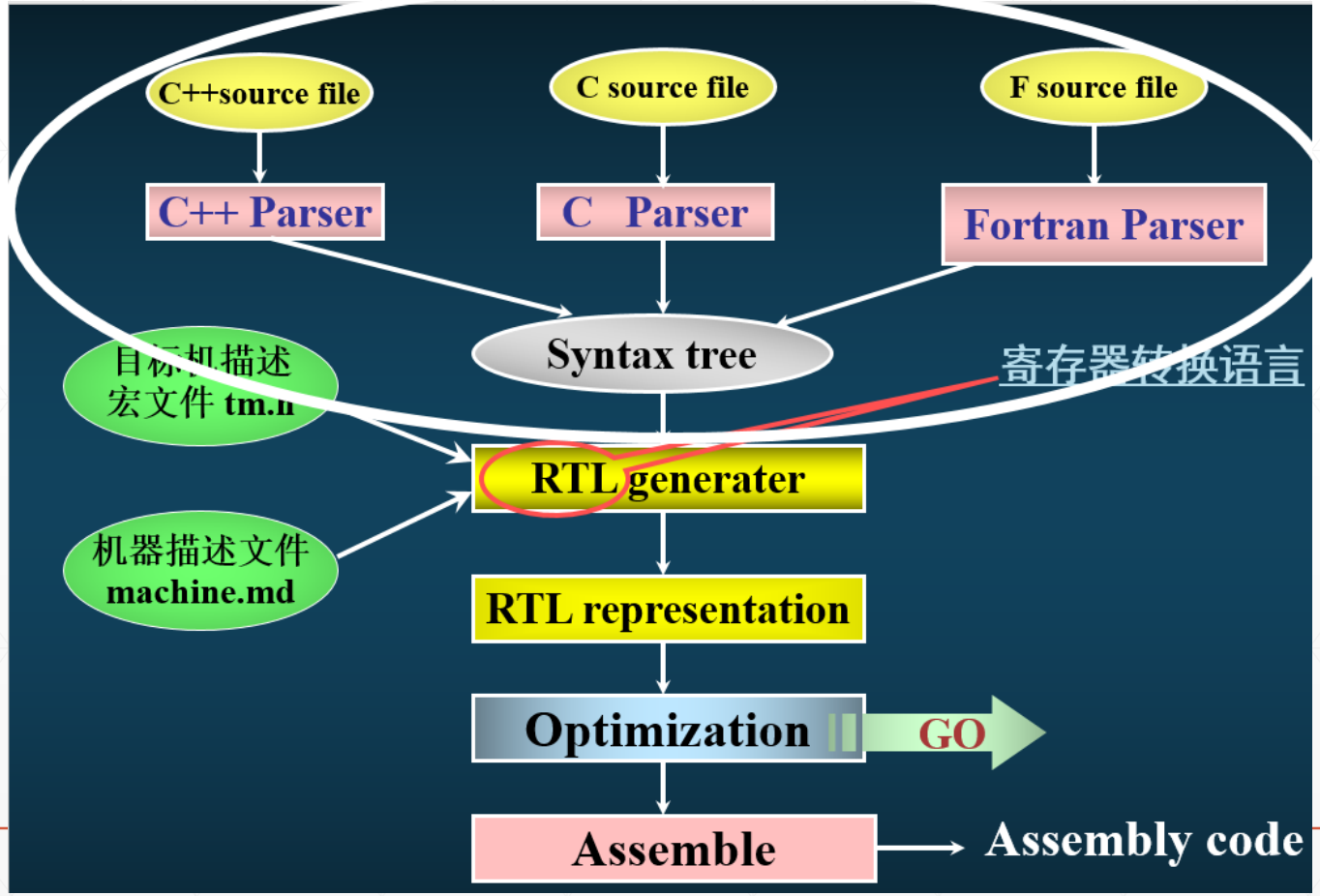
典型编译器结构

- Tiger compiler structure



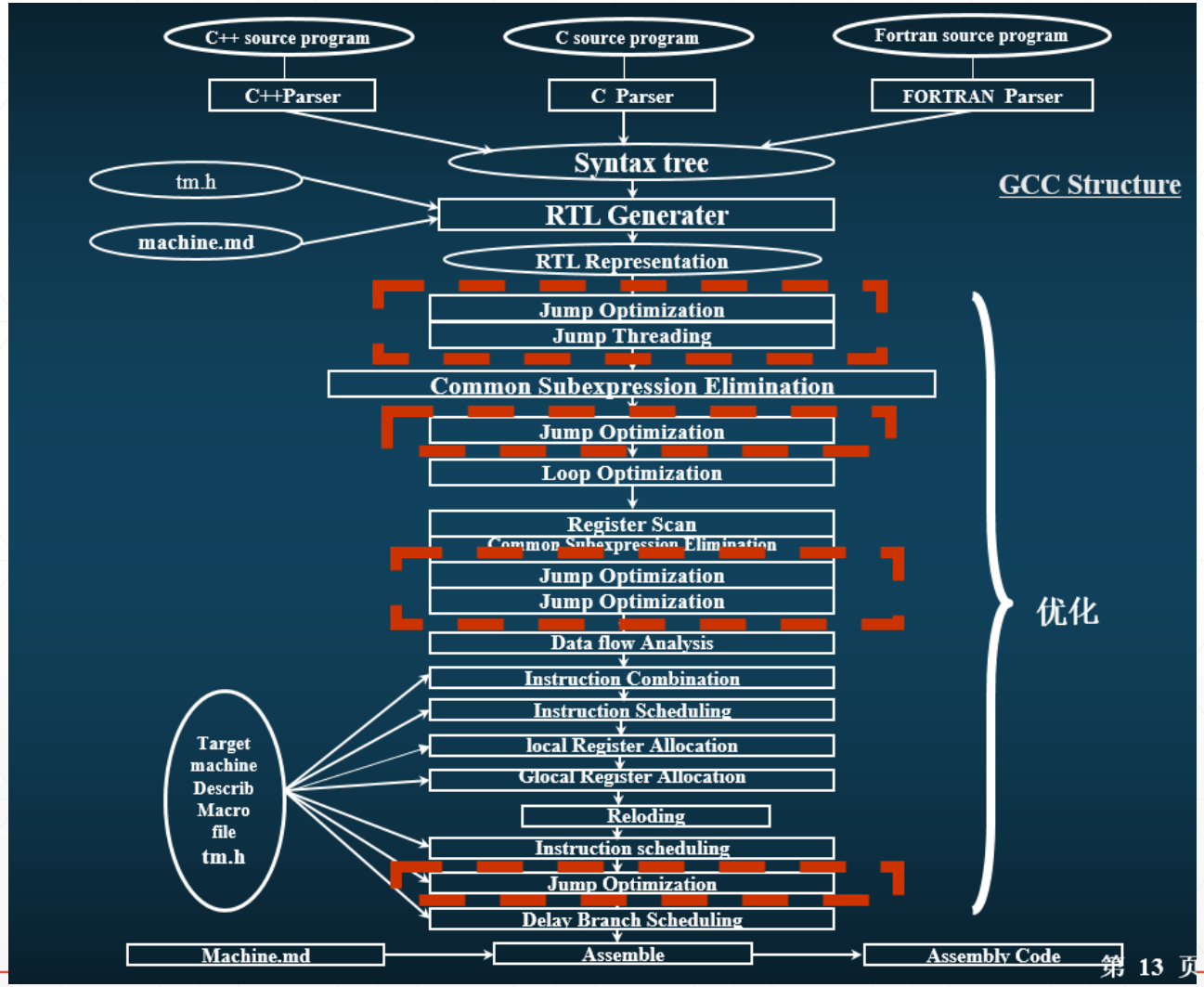
典型编译器结构

- GCC compiler structure



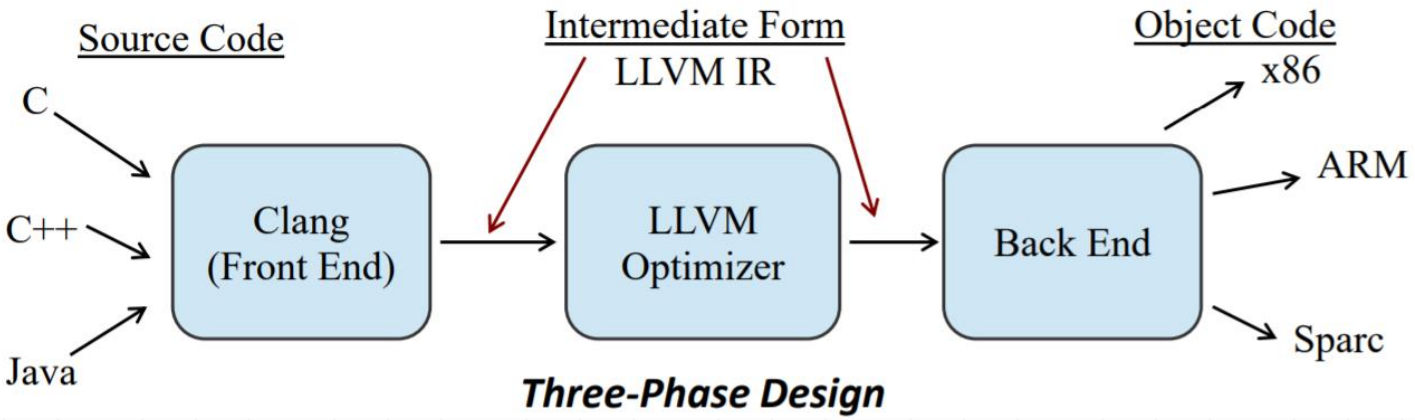
典型编译器结构

- GCC


C++ source program
C source program
Fortran source program
C++Parser
C Parser
FORTRAN Parser
Syntax tree
tm.h
machine.md
RTL Generater
GCC Structure
RTL Representation
Jump Optimization
Jump Threading
Common Subexpression Elimination
Jump Optimization
Loop Optimization
Register Scan
Common Subexpression Elimination
Jump Optimization
Jump Optimization
Data flow Analysis
Instruction Combination
Instruction Scheduling
local Register Allocation
Glocal Register Allocation
Reloding
Instruction scheduling
Jump Optimization
Delay Branch Scheduling
Target machine Describ Macro file tm.h
优化
Machine.md
Assemble
Assembly Code

典型编译器结构

- LLVM编译器结构

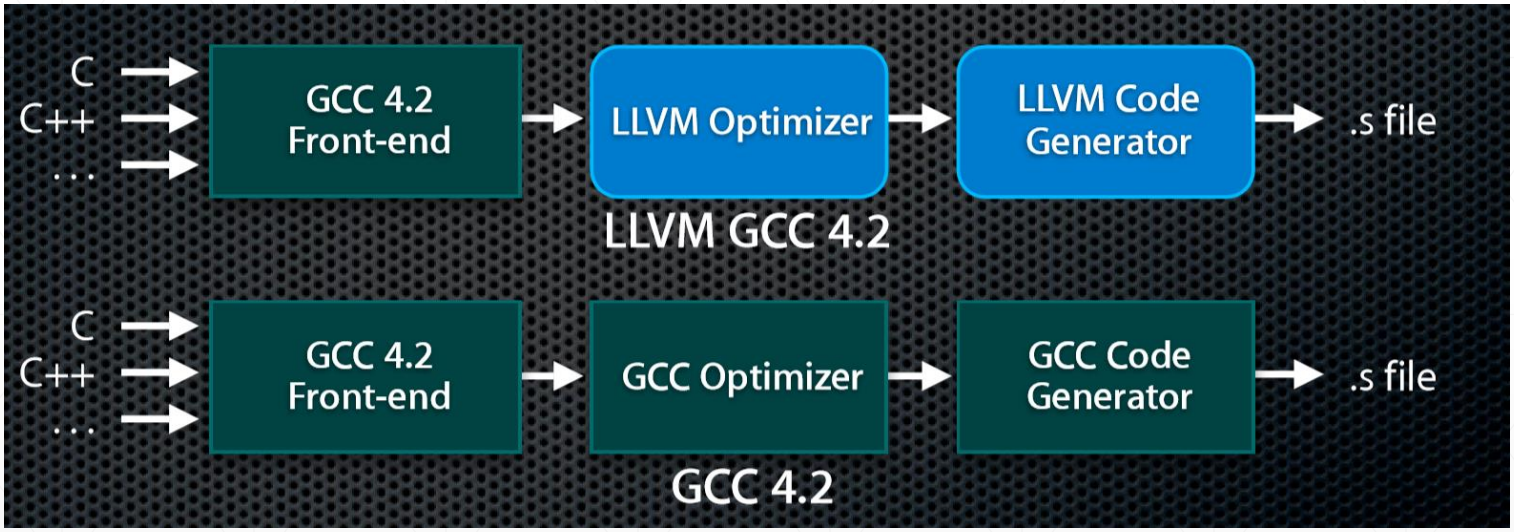


A research project at the University of Illinois

https://llvm.org/

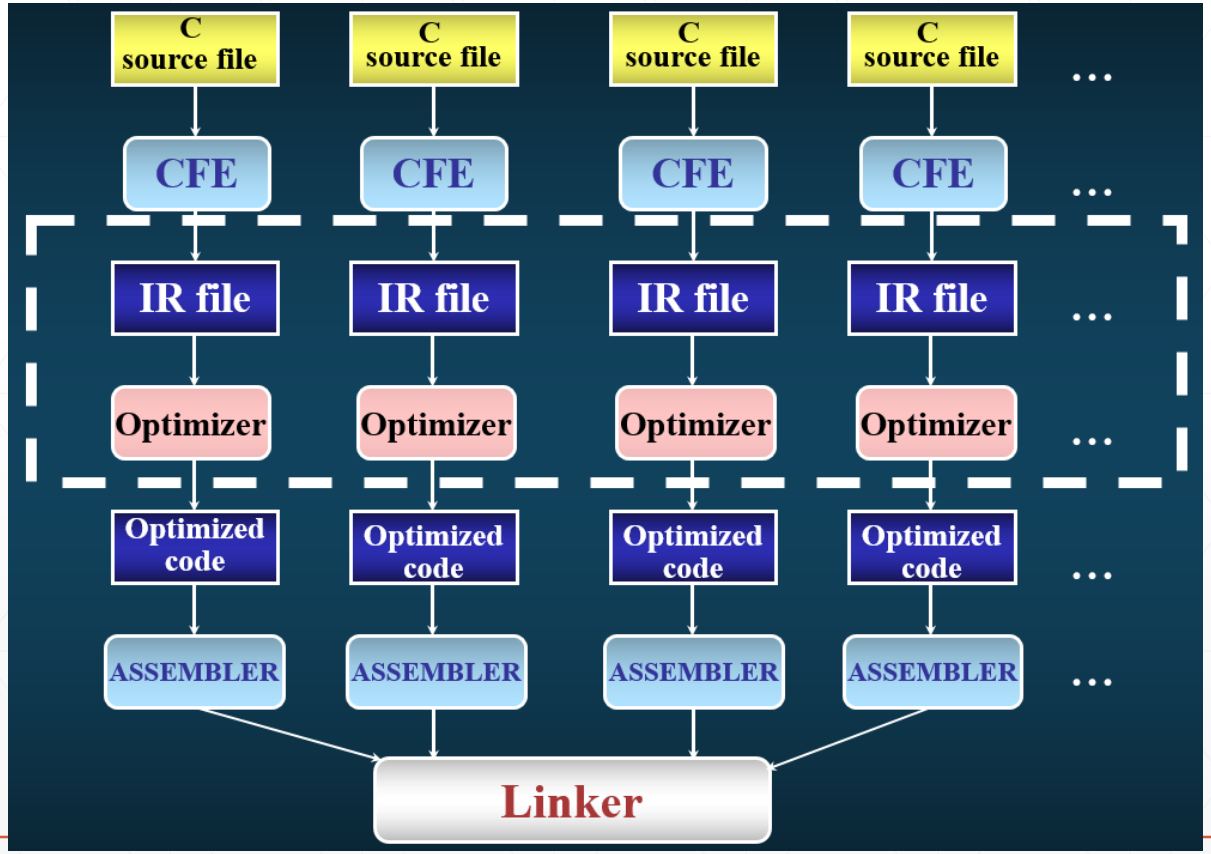
典型编译器结构

- LLVM编译器结构



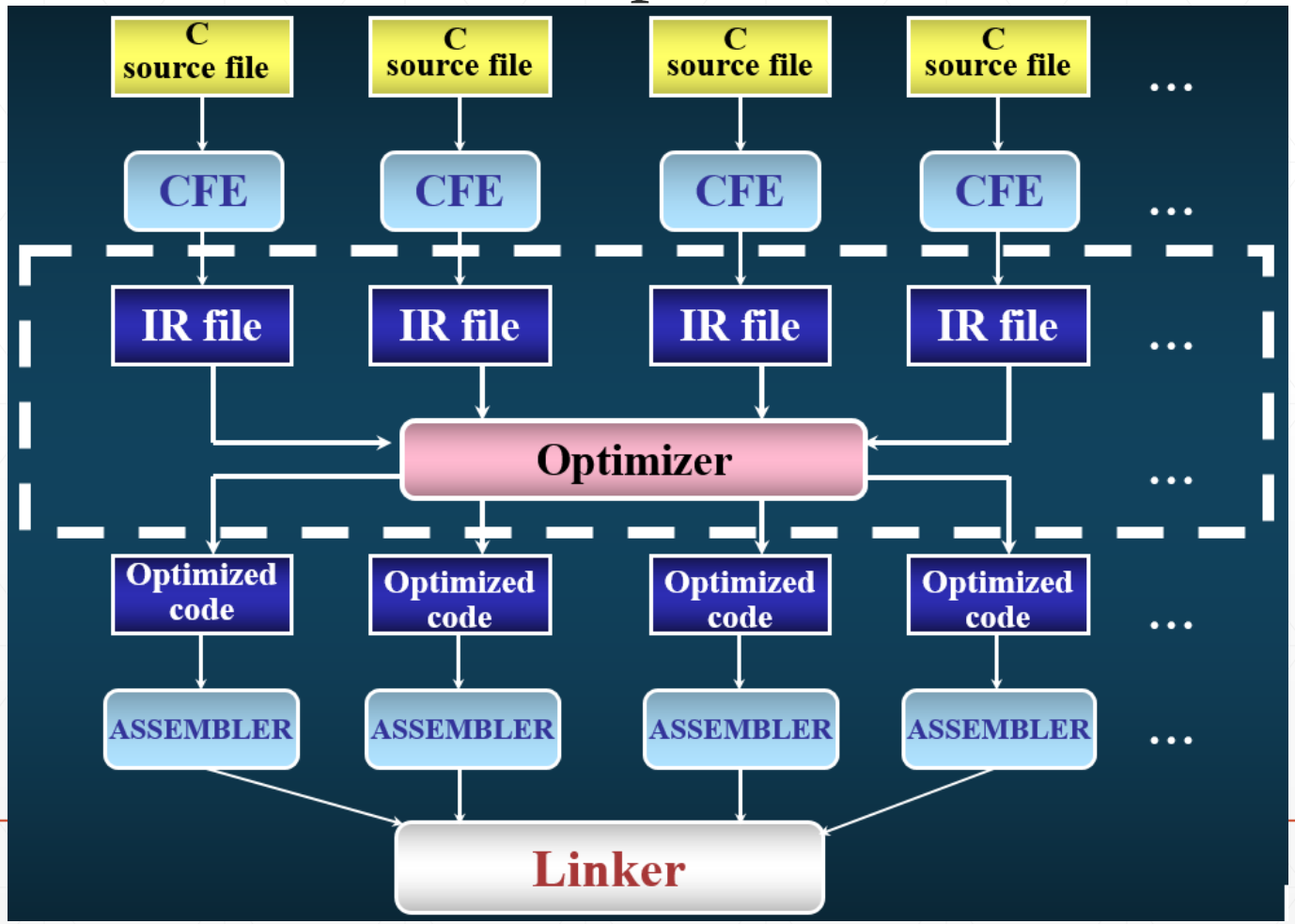
典型编译器结构

- StarCore Non Global Optimization



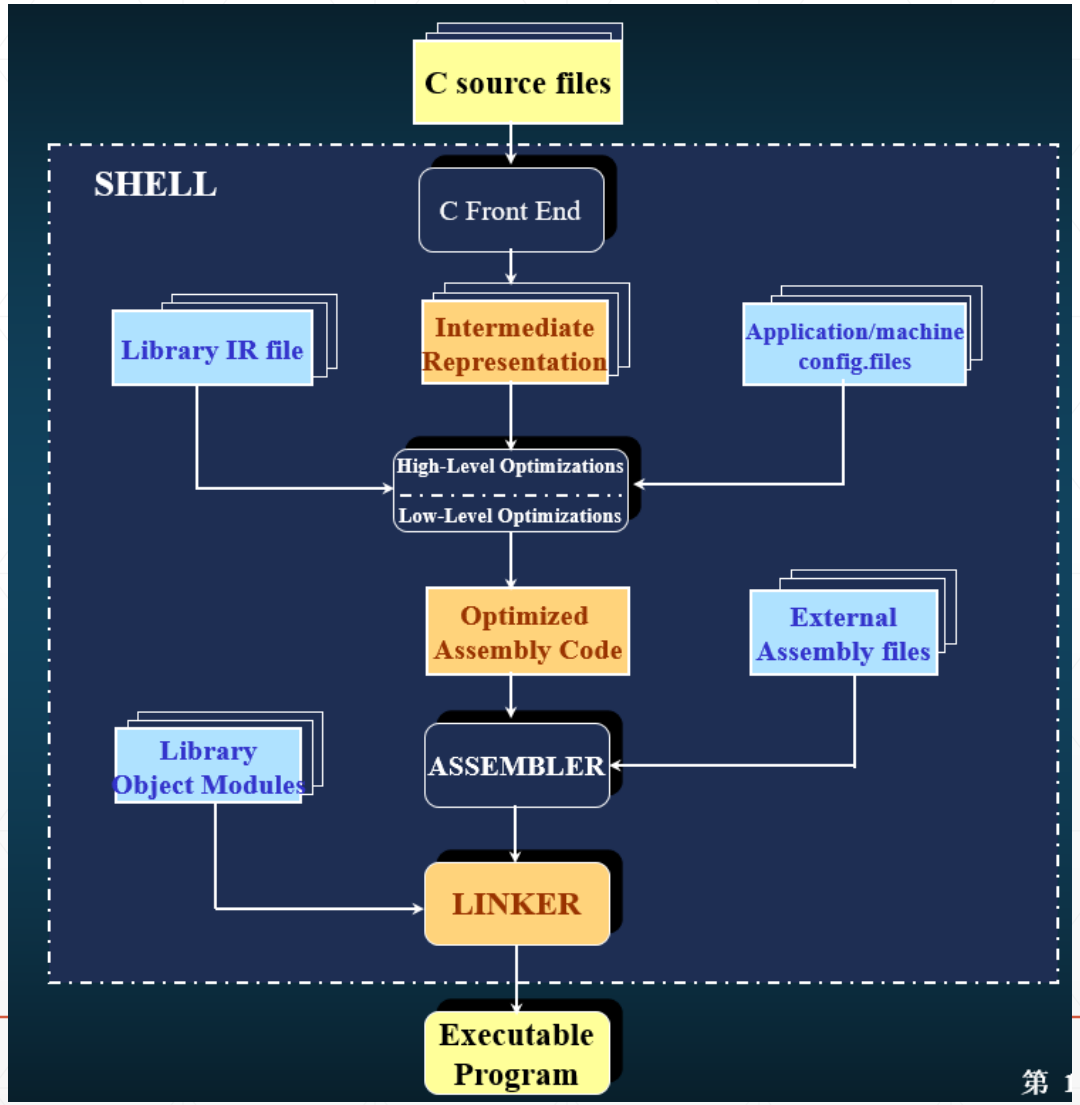
典型编译器结构

- StarCore Global Optimization



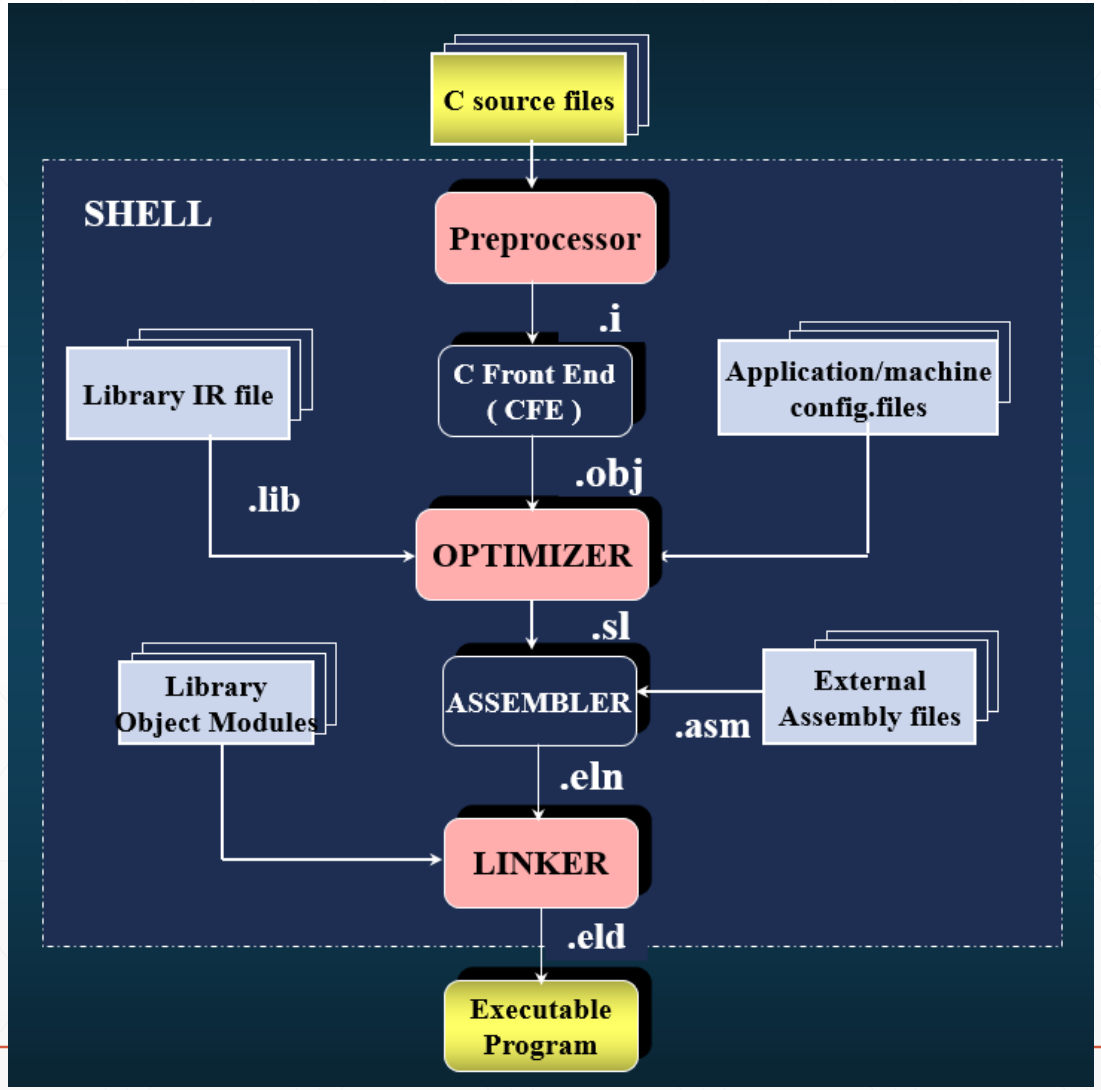
典型编译器结构

- SCC


C source files
SHELL
C Front End
Library IR file
Intermediate Representation
Application/machine config.files
High-Level Optimizations
Low-Level Optimizations
Optimized Assembly Code
External Assembly files
Library Object Modules
ASSEMBLER
LINKER
Executable Program

典型编译器结构

- SCC


C source files
SHELL
Preprocessor
.i
Library IR file
C Front End
(CFE)
Application/machine
config.files
.obj
.lib
OPTIMIZER
.sl
Library
Object Modules
ASSEMBLER
External
Assembly files
.asm
.eln
LINKER
.eld
Executable
Program

编译器的构造

- 构造要素

源语言

目标语言

编译方法、技术与工具

- 编译程序的生成
  - 使用编程语言；
  - 移植方式；
  - 自编译（用源语言作为宿主语言）；
  - 自动生成。

编译器的构造