



School of Computer Science & Technology
Harbin Institute of Technology

编译原理

Compiler Principles and Techniques

主讲：韩希先 教授 博士生导师

Email: hanxx@hit.edu.cn



课程性质与特点

- 课程性质

- 技术基础

- 基础知识要求

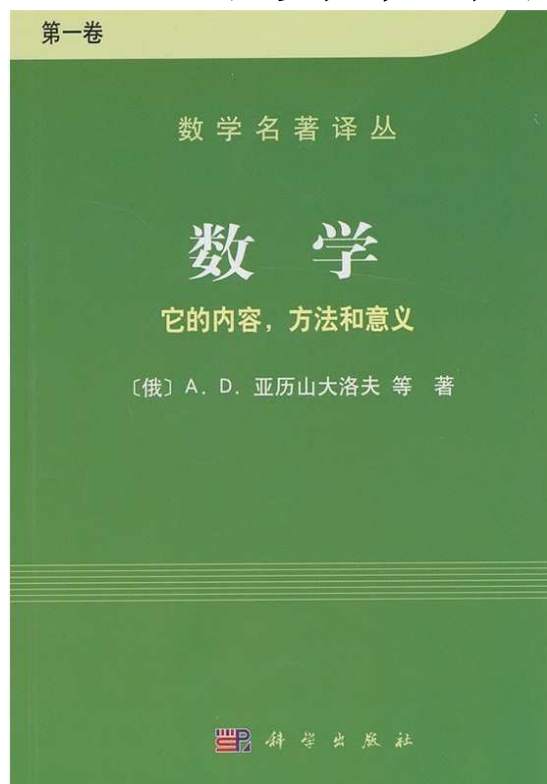
- 高级程序设计语言，数据结构与算法，形式语言与自动机

- 主要特点

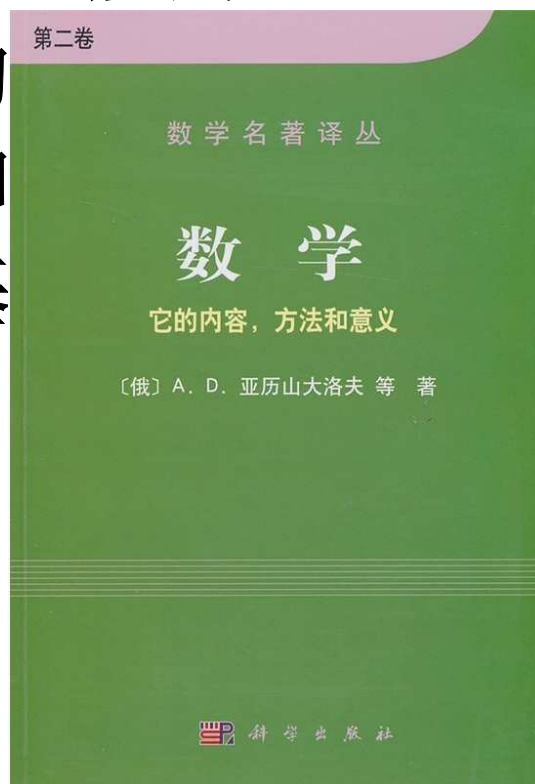
- 既有理论，又有实践
 - 面向系统设计
 - 涉及程序的自动生成技术

基础理论的重要性

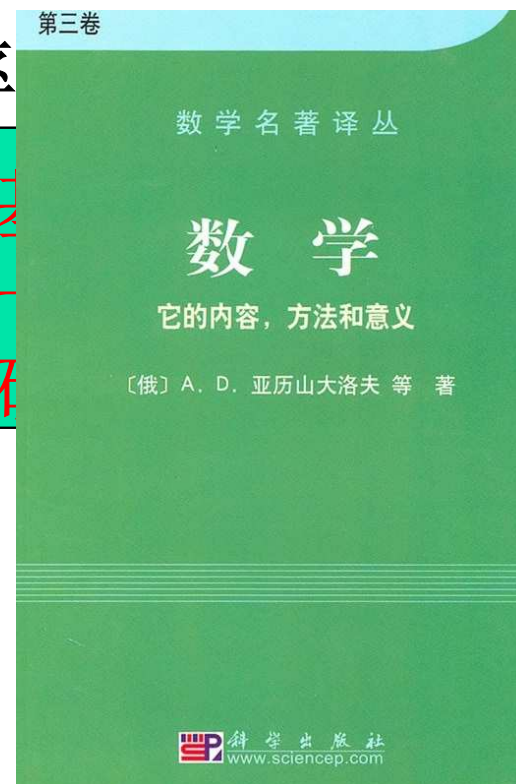
计算机内功的修炼



出的
构和
论基

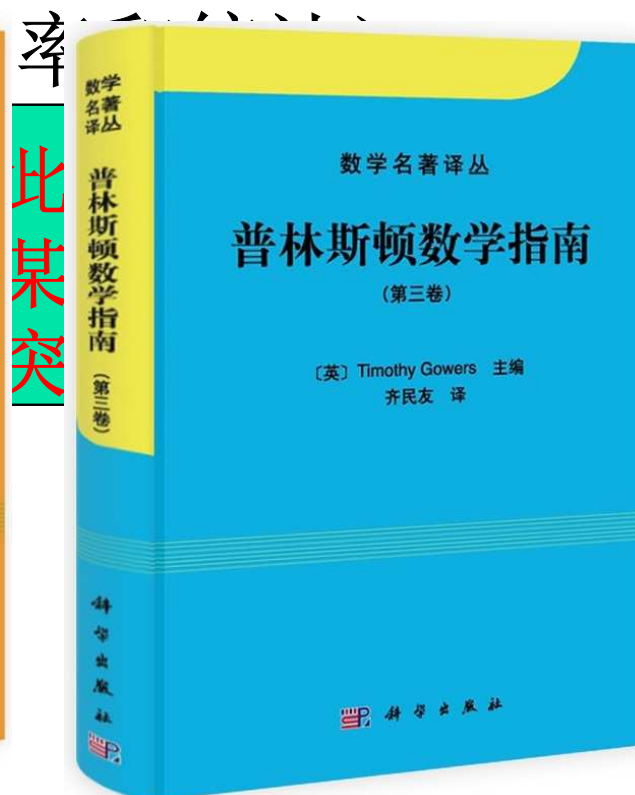
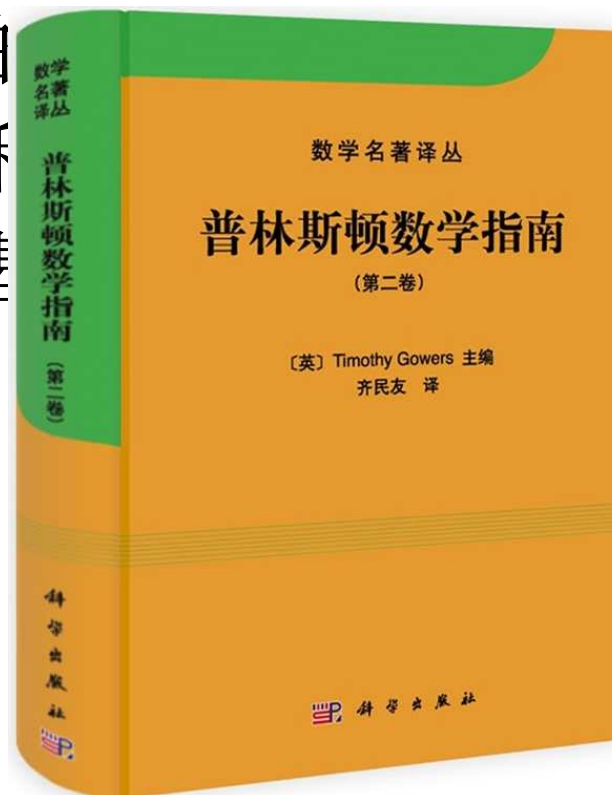
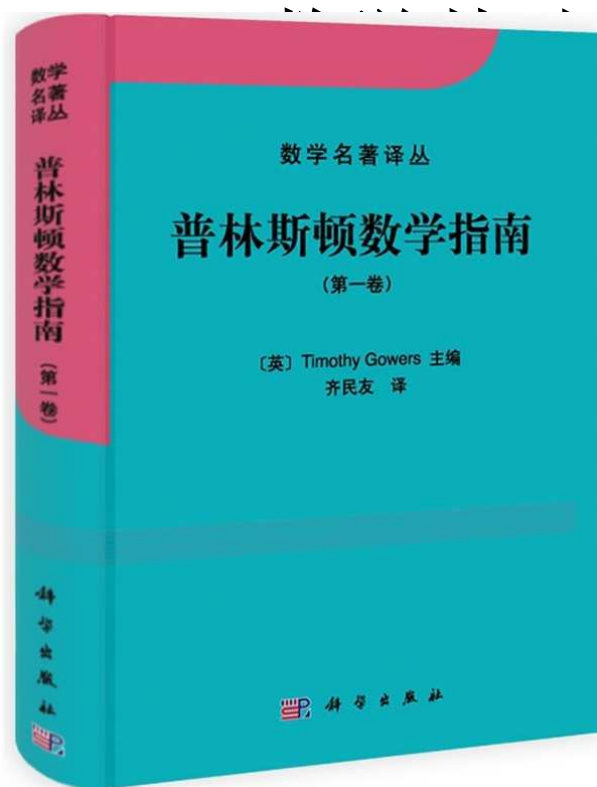


既率
在此
某
突破



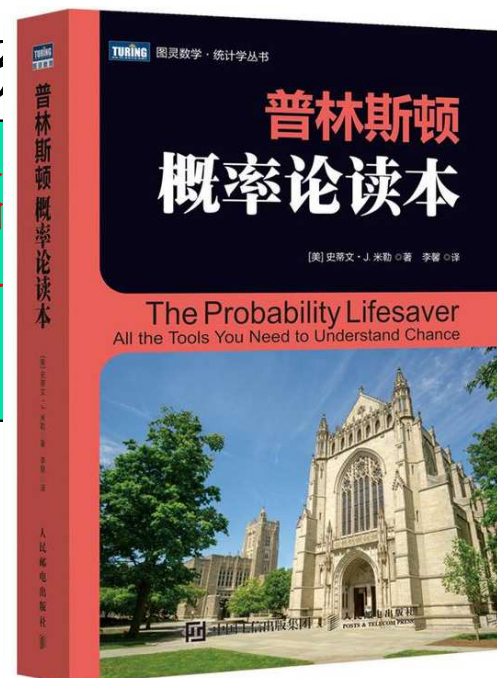
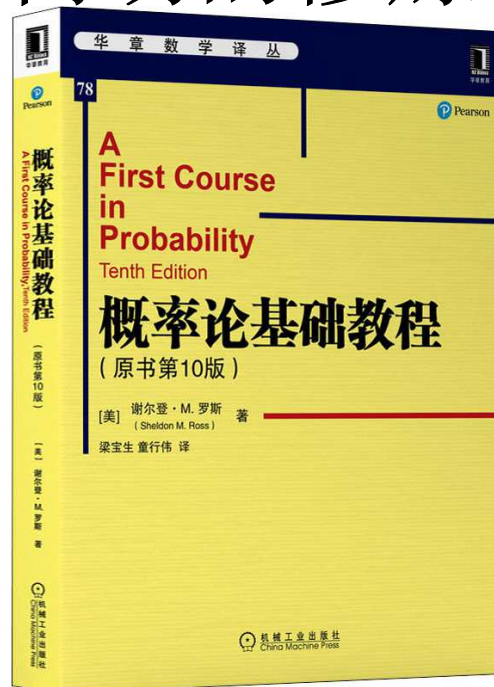
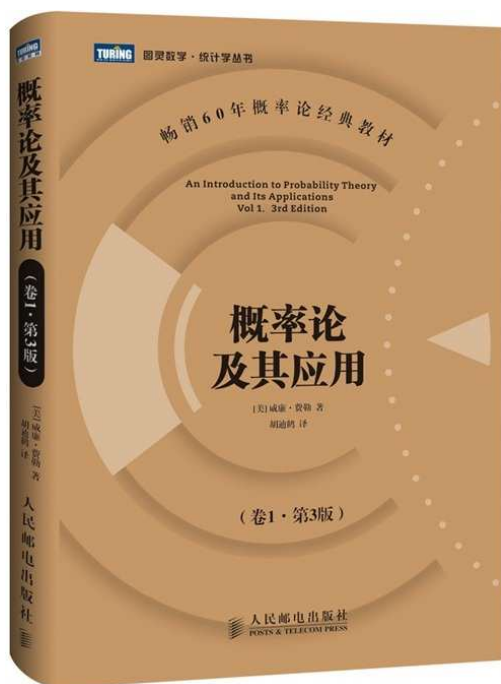
基础理论的重要性

■ 计算机内功的修炼



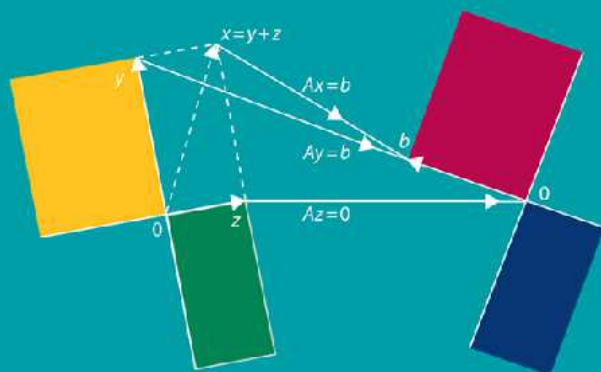
基础理论的重要性

计算机内功的修炼



基础理论的重要性

Introduction to
LINEAR ALGEBRA
FIFTH EDITION



GILBERT STRANG

提炼

(一)

基础

积累

LINEAR ALGEBRA
and Learning
from Data



GILBERT STRANG

基础理论的重要性



的修炼
积累（
算法基
础的积



统计)

出上寻
点作

- **1729**：哈代-拉玛努金数，“在所有能用两种方法写成两个立方和的数中，它最小。”
($1729=12^3+1^3=10^3+9^3$)
- 是否有一个数有至少十种不同方式写成四个不同正整数的立方和
- 考虑 $[1, 1000]$ 中的正整数集合



教学目的——《编译原理》是一门非常好的课程

- **Alfred V.Aho: 编写编译器的原理和技术具有十分普遍的意义，以至于在每个计算机科学家的研究生涯中，本课程中的原理和技术都会反复用到。**



教学目的——《编译原理》是一门非常好的课程

- 本课程将兼顾语言描述方法、设计与应用
 - 能形式化就能自动化（抽象→符号化→机械化）
 - 可以使学生对程序设计语言具有更加深刻的理解
 - 体验实现自动计算的乐趣



教学目的——《编译原理》是一门非常好的课程

- 涉及的是一个比较适当的抽象层面上的数据变换（既抽象又实际，既有理论又有实践）
- 一个相当规模的系统的设计
 - 总体结构
 - 若干具体的表示和变换算法



教学目的(续)

- 在**系统级**上认识算法和系统的设计
 - 具有把握系统的能力
 - 局部最优vs.全局最优(木桶效用)
 - “自顶向下”和“自底向上”的系统设计方法, 对其思想、方法、实现的全方位讨论
- 进一步培养 “**计算思维能力**”
 - 深入理解软件系统的非物理性质
 - 培养抽象思维能力和逻辑思维能力
 - 训练对复杂数据结构的设计和操纵能力



计算思维能力

- **定义：运用计算机科学的基础概念进行问题求解、系统设计、以及人类行为理解等涵盖计算机科学之广度的一系列思维活动**
- **2006年，周以真(Jeannette M. Wing, CMU教授)在Communications of the ACM提出**

计算思维能力

- 日常生活和计算机科学
 - 晚上把第二天上课需料放进书包
 - 数据缓存和预取操作
 - 钱包丢了，沿走
 - 树和图的回溯
 - 超市买东西，决定
 - 大规模分布式系统的
 - 家里停电了，但是自己的手机还可以用
 - 关键系统的冗余性设计



计算思维能力

- **求解问题的一般步骤**
 - **问题是否可解**
 - **问题是否难解**
 - **非难解问题需要的是精确解还是近似解**
 - **编写程序**

计算思维能力

■ 问题是否可解

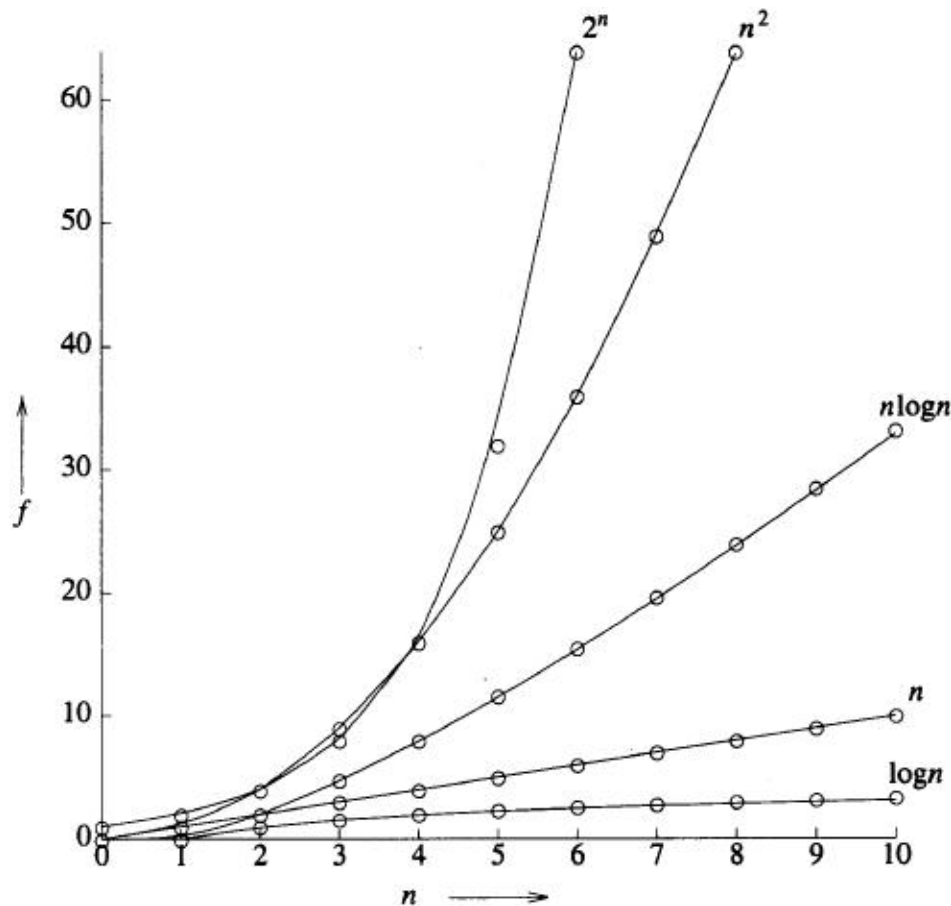
- 图灵停机问题：不存在一个程序，可以判断任何程序在给定输入上是否结束

```
bool Good_algo(char* program,
char* input)
{
    if(<program> halts on
<input>)
        return true;
    return false;
}
```

```
bool Bad_algo(char* input)
{
    if(Good_algo(Bad_algo, input) )
    {
        while(1);    // loop forever!
        return false; // never get here
    }
    else
        return true;
}
```

计算思维能力

■ 问题是否难解



该类问题称为：
NP-完全问题

该问题

个城市推

一个包含所有n个城市的具

法，时间复杂度 $O(n!)$

比较：

$n, n^2, n^3, 2^n, n!$



计算思维能力

■ 问题是否难解

- 旅行商问题：有一个推销员，要到 n 个城市推销商品，他要找出一个包含所有 n 个城市的具有最短路程的环路
- 枚举办法 $O(n!)$ 到底有多慢：
IBM Roadrunner超级计算机，1.33亿美元，运算速度1457万亿次/秒，当 $n=33$ 时，枚举办法的计算时间
- 28000亿年（宇宙137亿年，地球45亿年）



计算思维能力

- **问题是否难解**

- **近似/随机算法: 爬山算法、模拟退火、遗传算法、蚁群算法、单纯形算法、分值切割法**
- **1971年, 兰德公司, $n=49$**
- **1975年, 意大利研究团队, $n=67$**
- **1987年, 德国研究团队, $n=666$**
- **1998年, 贝尔实验室, $n=13509$**
- **2006年, 贝尔实验室, $n=85900$**
- **全球城市总数: 1904711, 至今未解**



计算思维能力

- **非难解问题**

- **存在多项式时间算法**
- **排序问题**
- **最近邻问题**
- **图遍历问题**
- **等等**
- **准确解与近似解问题**



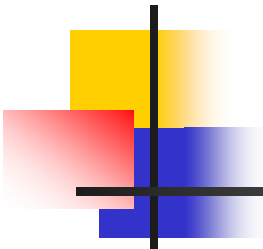
计算思维能力

- **编写程序**
 - **最后一步**
 - **不能满足于这一步**



教学目的(续)

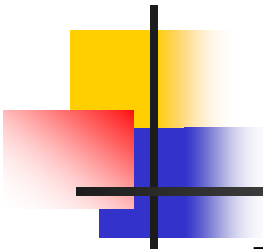
- **计算机专业最为恰当、有效的知识载体之一**
- **综合运用下列课程所学知识**
 - **高级程序设计语言**
 - **汇编语言**
 - **集合论与图论**
 - **数据结构与算法**
 - **计算机组成原理**
 - **算法设计与分析**
 - **形式语言与自动机**



教学要求——课程要求

■ 知识要求

- **掌握编译程序的总体结构**
- **编译程序各个组成部分的任务**
- **编译过程各个阶段的工作原理**
- **编译过程各个阶段所要解决的问题及其采用的方法和技术**



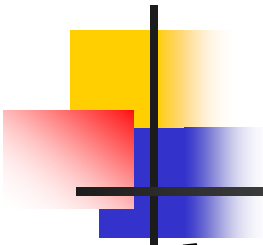
教学要求——实验要求

- **实验形式**

- **分析、设计、编写、调试、测试程序**
- **撰写实验报告**

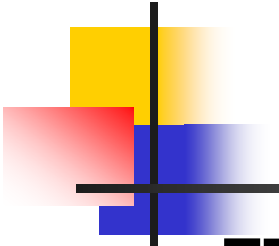
- **实验内容**

- **词法分析器的设计与实现**
- **语法分析器的设计与实现**



教学要求——实验目的

- 实验贯穿于理论、抽象和设计过程；
- 实验对软件的设计和实现、测试原理和方法起示范作用；
- 实验不仅仅是对理论的验证，重要的是**技术训练和能力培养**，包括动手能力、分析问题、解决问题能力、表达能力、写作能力等的培养；
- 教学活动是教师和学生不断交流的过程，**实验是实现这个过程桥梁**，可以弥补课堂教学的不足，加深理论过程的理解，启发学生深入思考，敢于创新，达到良好的理论联系实际的教学效果。



教学要求——考试要求

■ 题型

- 简答、证明、论述、设计等

■ 重点和难点

- 会在各章的开始点明

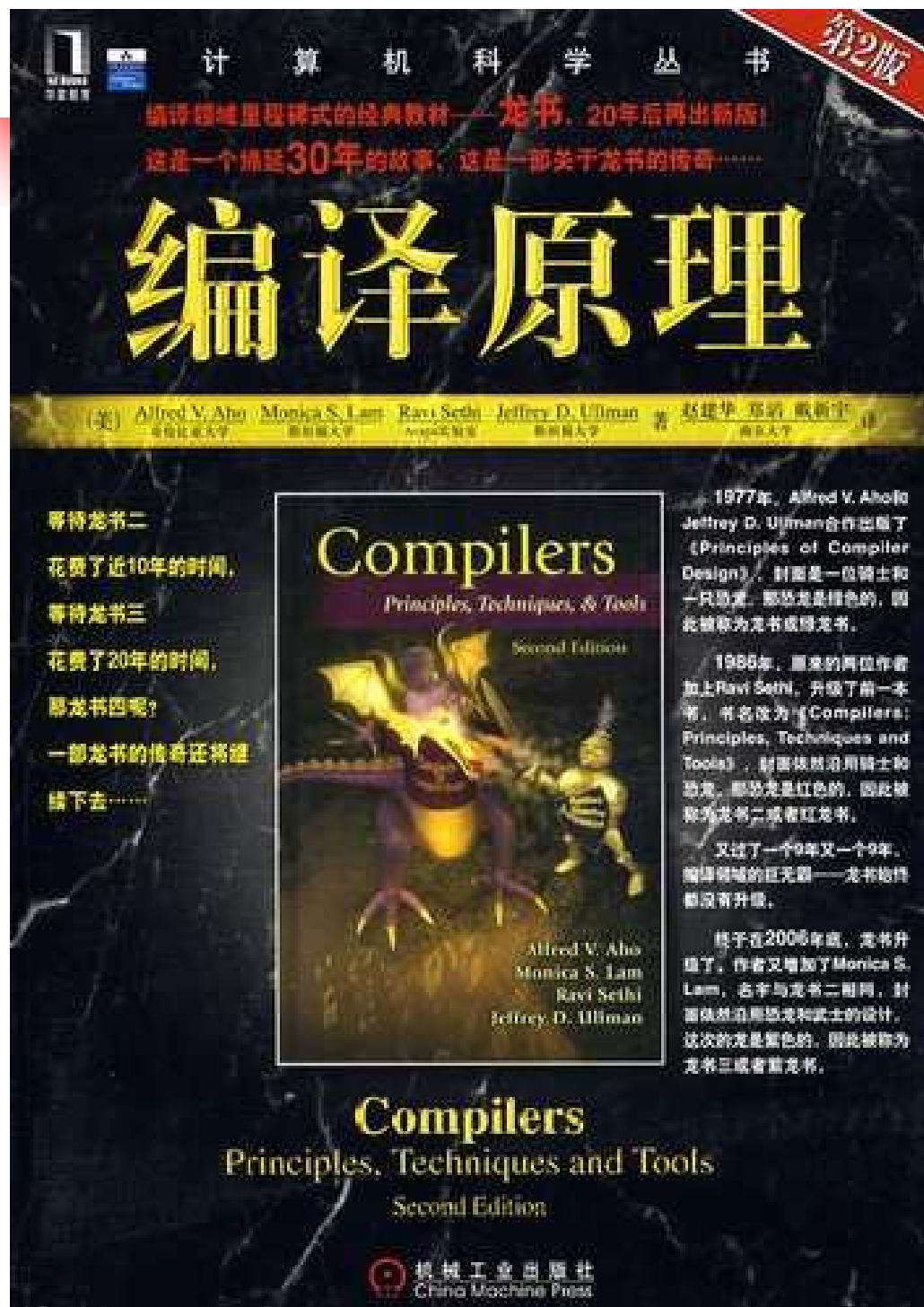
■ 考试权重

- 平时出勤(随堂测试)占10%
- 实验课程占20%
- 期末考试占70%



主要内容

1. 引论
2. 高级语言及其文法
3. 词法分析
4. 自顶向下的语法分析
5. 自底向上的语法分析
6. 语法制导翻译与属性文法
7. 中间代码生成
8. 符号表管理
9. 运行时的存储组织
10. 代码优化
11. 代码生成



5等教育本科国家级规划教材

面向工程教育的本科计算机类专业系列教材

Compiler Principles
and Techniques

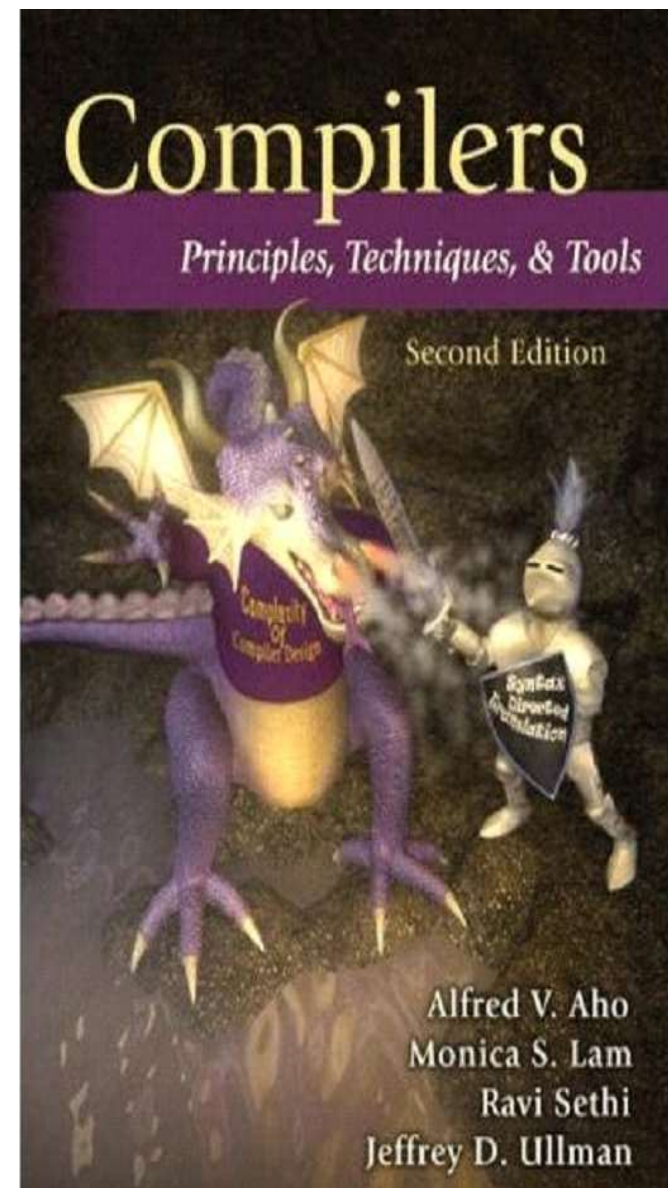
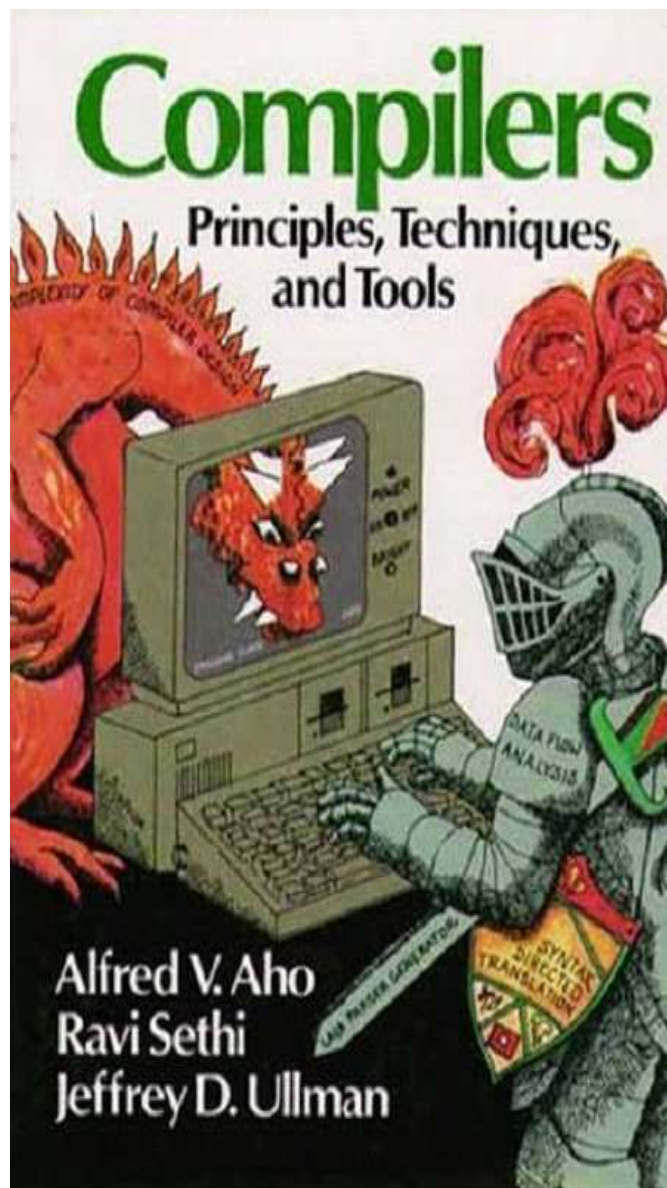
编译原理

(第2版)

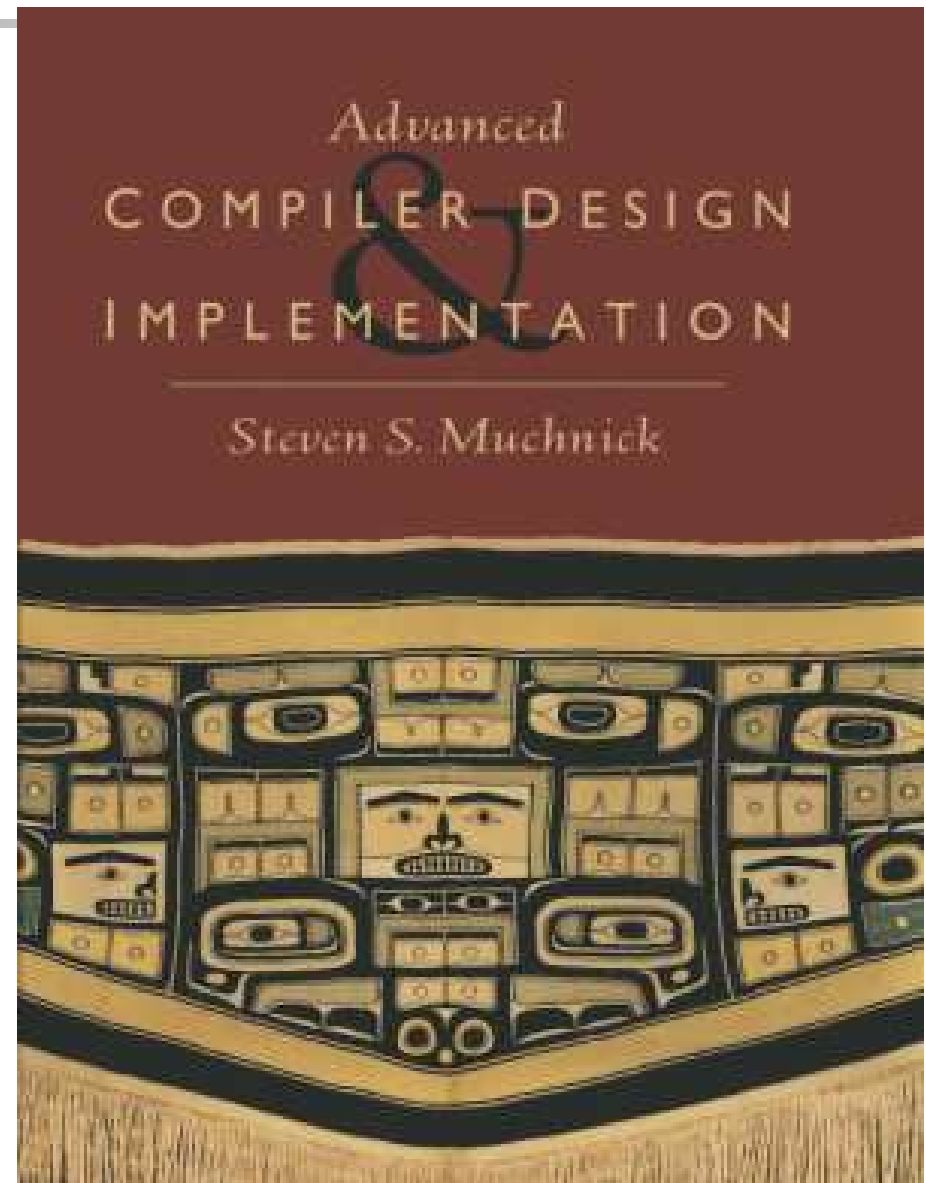
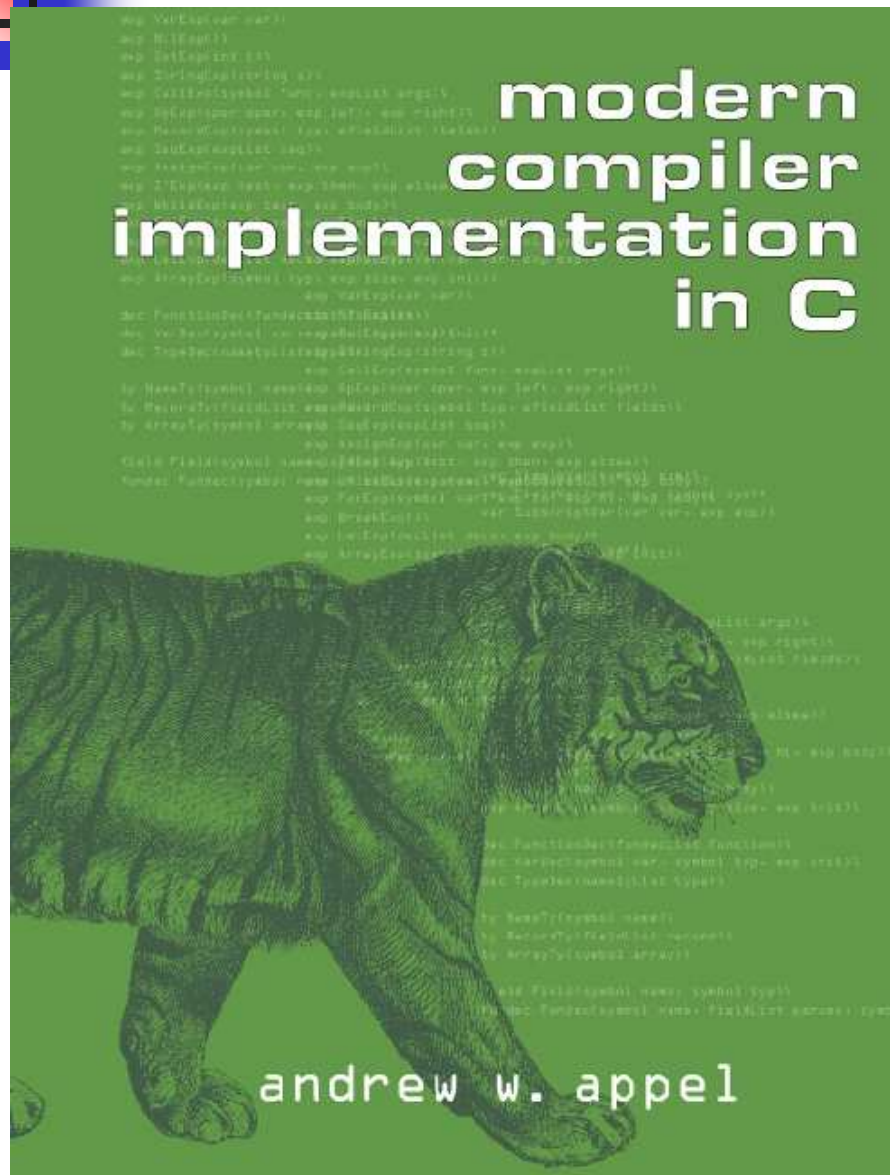
蒋宗礼 姜守旭 编著

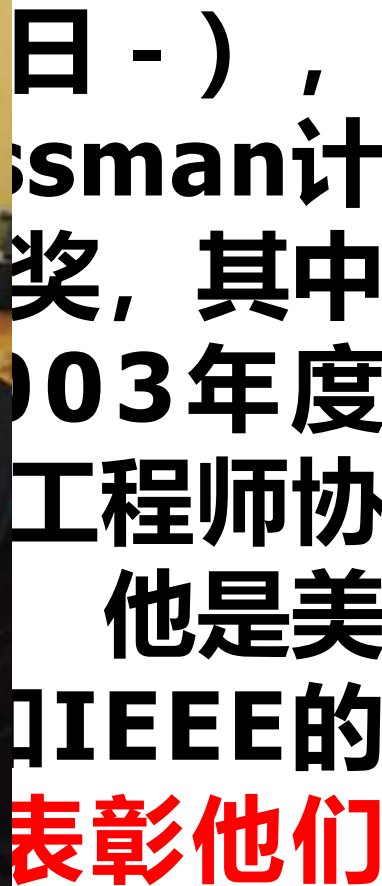
出版社

龙书




虎书和鲸书





- 在编程语言实现的基础算法和理论上做出的贡献)。

- 
- A close-up portrait of a middle-aged man with short, wavy white hair and a full white beard and mustache. He is smiling, showing his teeth. He is wearing a light blue collared shirt. The background is a soft, out-of-focus mix of light blue and white.

ance公司的首
的Stanfordc
（名誉退休）
数据库理论、数
信息基础软件
家工程院的院
Istrom奖和
年图灵奖获得

2020年图灵奖获得

Jeffery D. Ullman





第一章 引论

重点： 教学目的，教学要求，学习方法，课程的基本
内容，编译系统的结构，编译程序的生成。
难点： 编译程序的生成。





第1章 引论

1.1 程序设计语言

1.2 程序设计语言的翻译

1.3 编译程序的总体结构

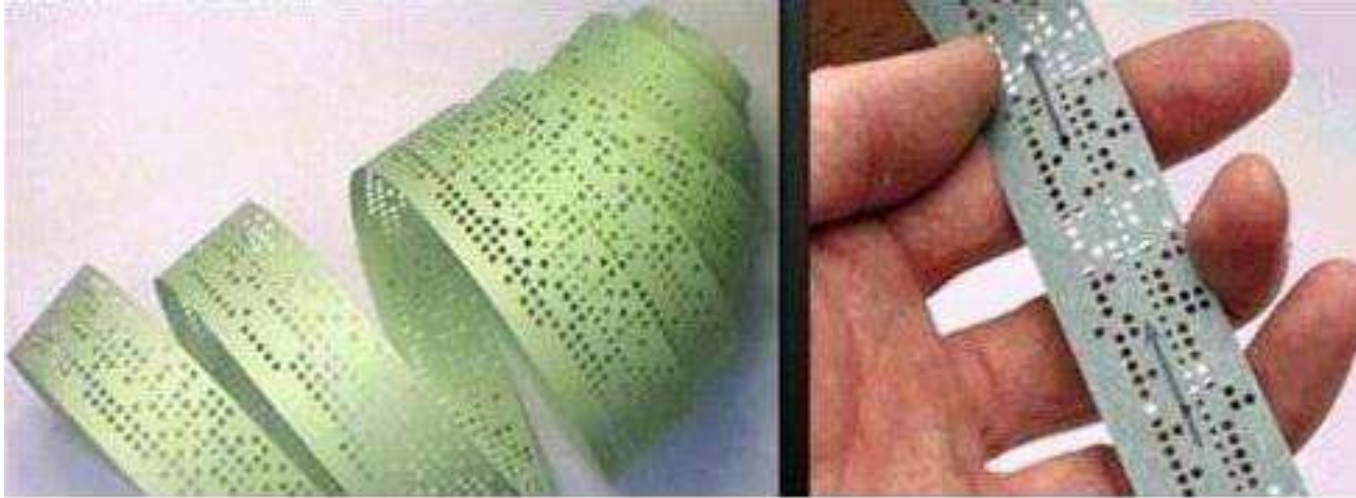
1.4 编译程序的组织

1.5 编译程序的生成

1.6 本章小结

1.1 程序设计语言

■ 机器语言(Machine Language) - 每一个



1011 1000 0010 1011 0001 0101
(B82B15)

1000 1110 1101 1000 (8ED8)

0000 0000 0000 0000

0000 0000 0010
(00)

0000 0000 0000

0000 (03C8)

0000 0011 1100 1011 (03CB)

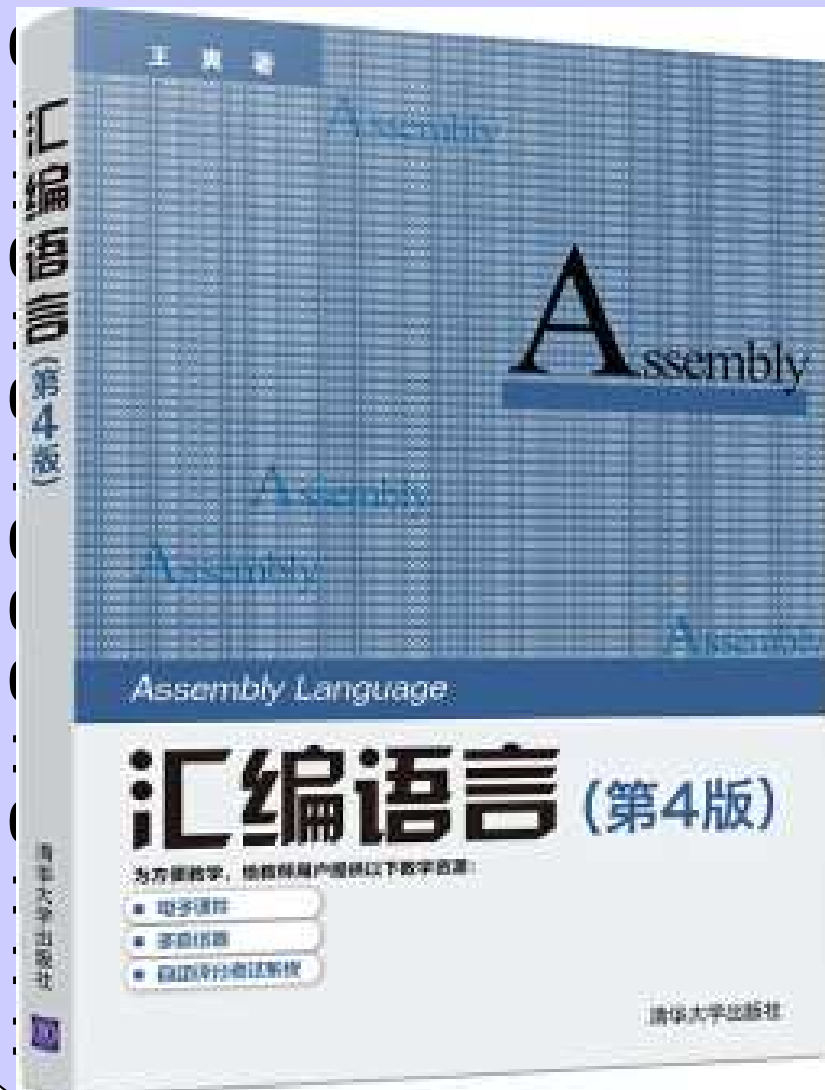
1000 1011 0000 1110 0000 0100

0000 0000 (8B0E0400)

1011 1000 0000 0000 0100 1100
(B8004C)

1100 1101 0010 0001 (CD21)

1011 1000 0010 1011 0001



```
assume cs:code, ds:data
data segment
```

```
    dw 1234h,5678h
```

```
data ends
```

```
code segment
```

```
start:  mov ax, data
```

```
        mov ds, ax
```

```
        mov ax, ds:[0]
```

```
        mov bx, ds:[2]
```

```
        mov cx, 0
```

```
        add cx, ax
```

```
        add cx, bx
```

```
        mov ds:[4], cx
```

```
        mov ax, 4c00h
```

```
        int 21h
```

```
code ends
```

```
end start
```




**assume cs:code, ds:data
data segment**

dw 1234h,5678h

data ends

code segment

**start: mov ax, data
mov ds, ax
mov ax, ds:[0]
mov bx, ds:[2]
mov cx, 0
add cx, ax
add cx, bx
mov ds:[4], cx
mov ax, 4c00h
int 21h**

**code ends
end start**

语言

Level Language)

近

运算

RAN

**int main
{**

**int a,b,c;
a=1234h;
b=5678h;
c=a+b;
return 0;**

}

1.2 程序设计语言的翻译

■ 翻译程序(Translator)

将某一种语言描述的程序(源程序——Source Program)翻译成**等价的**另一种语言描述的程序(目标程序——Object Program)的程序。





1.2 程序设计语言的翻译

- **翻译程序(Translator)**

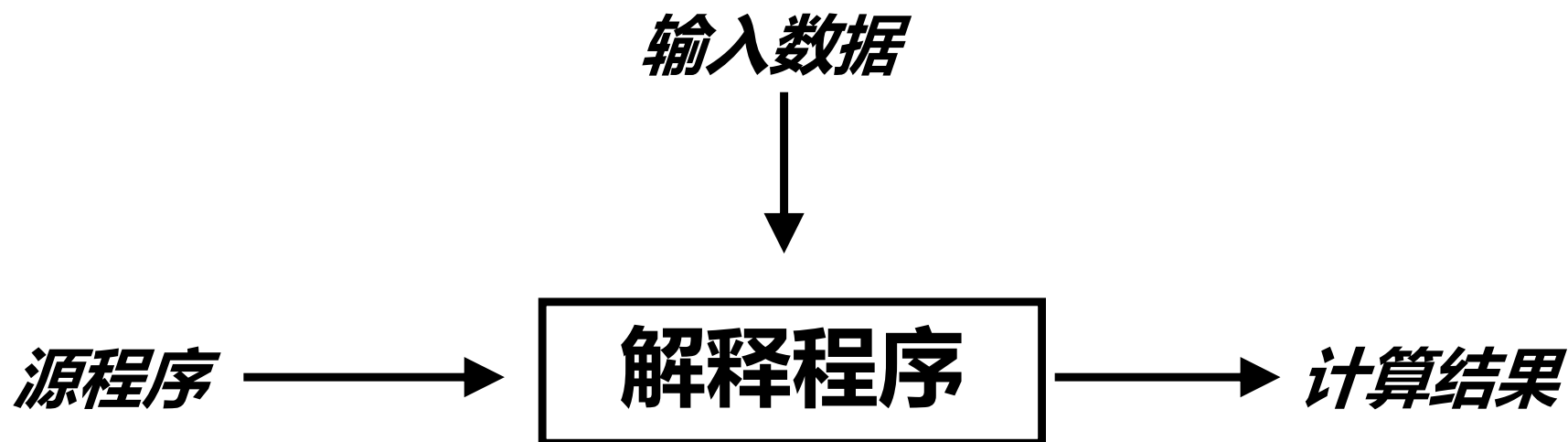
- **解释程序**

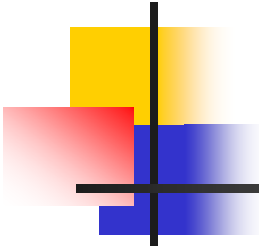
- **编译程序**

1.2 程序设计语言的翻译

■ 解释程序(Interpreter)

- 边解释边执行：不断读取源程序的语句，解释语句，读取此语句需要的数据，根据执行结果读取下一条语句，继续解释执行，直到返回结果
- 类似于自然语言翻译的同声传译





1.2 程序设计语言的翻译

- **编译程序(Compiler)**

- **将源程序完整地转换成机器语言程序或汇编语言程序，然后再处理、执行的翻译程序**
- **高级语言程序→汇编/机器语言程序**
- **类似于自然语言翻译的通篇笔译**



1.2 程序设计语言的翻译

SP → Compiler

S-Source

OP

O-Object

P-Program

Input → RS

RS-Run Sys.

支撑环境、
运行库等

Output

编译系统(Compiling System)

编译系统=编译程序+运行系统



1.2 程序设计语言的翻译

- 其它翻译程序:

- 汇编程序(Assembler)

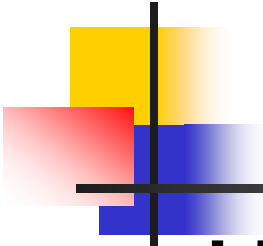
当源程序是汇编语言程序，目标程序是机器语言程序



1.2 程序设计语言的翻译

- 其它翻译程序：
 - 汇编程序(Assembler)
 - 交叉汇编程序(Cross Assembler)

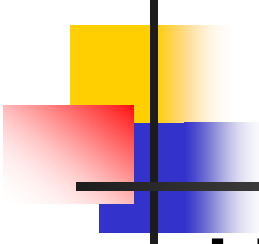
源程序是汇编语言程序，
目标程序是另一台机器
的机器语言程序



1.2 程序设计语言的翻译

- 其它翻译程序：
 - 汇编程序(Assembler)
 - 交叉汇编程序(Cross Assembler)
 - 反汇编程序 (Disassembler)

源程序是机器语言程序，
目标语言是汇编语言程序



1.2 程序设计语言的翻译

- **其它翻译程序：**

- **汇编程序(Assembler)**
- **交叉汇编程序(Cross Assembler)**
- **反汇编程序 (Disassembler)**
- **交叉编译程序 (Cross Compiler)**

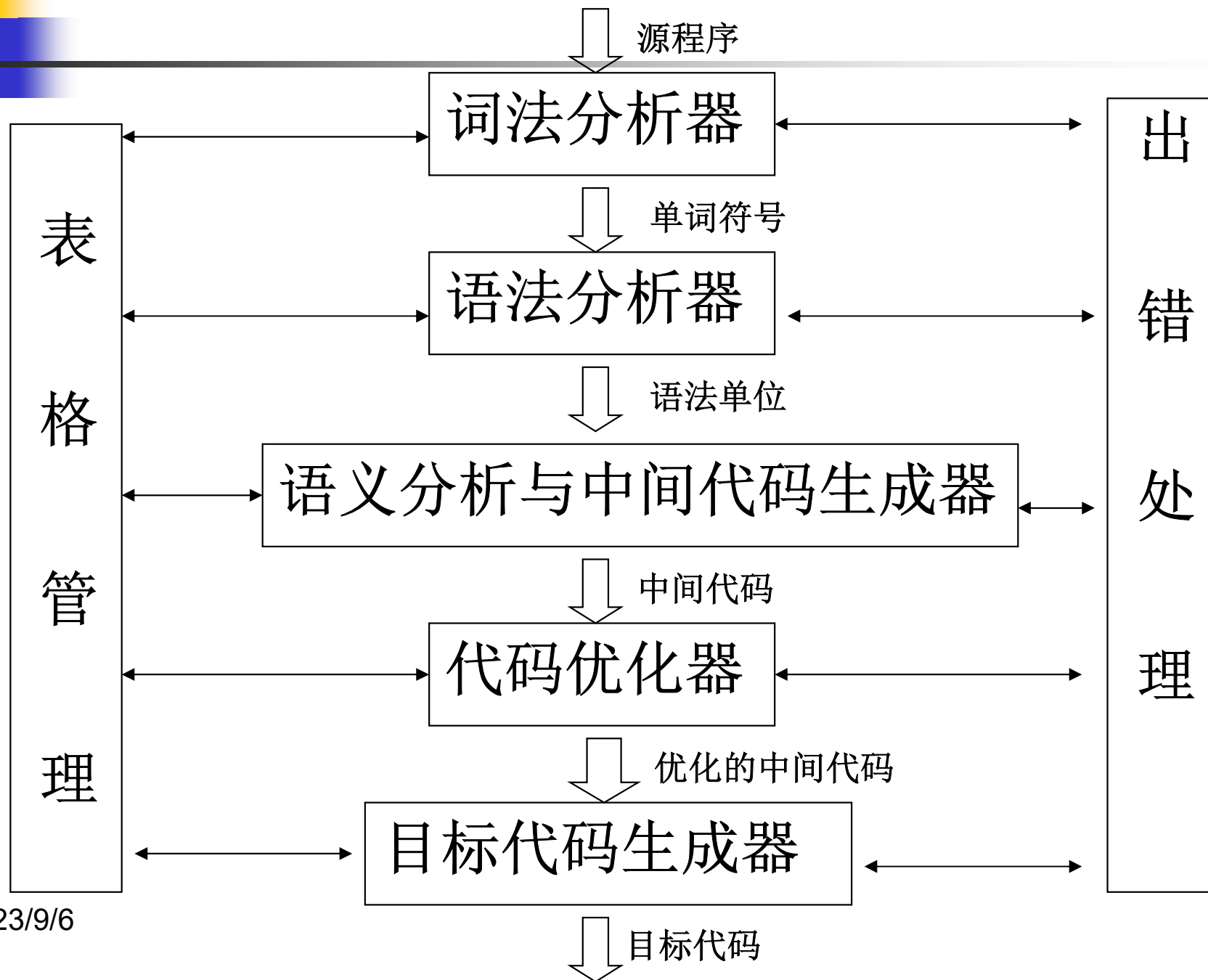
源程序是高级语言程序，
目标程序是另一台机器的
机器/汇编语言程序



1.2 程序设计语言的翻译

- **其它翻译程序：**
 - **汇编程序(Assembler)**
 - **交叉汇编程序(Cross Assembler)**
 - **反汇编程序 (Disassembler)**
 - **交叉编译程序 (Cross Compiler)**
 - **反编译程序 (Decompiler)**
 - **可变目标编译程序 (Retargetable Compiler)**
 - **并行编译程序 (Parallelizing Compiler)**
 - **诊断编译程序 (Diagnostic Compiler)**
 - **优化编译程序 (Optimizing Compiler)**

1.3 编译程序总体结构





1、词法分析

- **词法分析由词法分析器(Lexical Analyzer)完成，词法分析器又称为扫描器(Scanner)**
- **词法分析器从左到右扫描组成源程序的字符串，并将其转换成单词串；同时要：查词法错误，进行标识符登记（符号表管理）。**
- **输入：字符串**
- **输出：(种别码，属性值)——序对**
 - **属性值——token的机内表示**



1、词法分析

■ 例:

```
sum=(10+20)*(num+square);
```

结果

- (标识符, sum)
- (赋值号, =)
- (左括号, (
- (整常数, 10)
- (加号, +)
- (整常数, 20)
- (右括号,))
- (乘号, *)
- (左括号, (
- (标识符, num)
- (加号, +)
- (标识符, square)
- (右括号,))
- (分号, ;)



2、语法分析

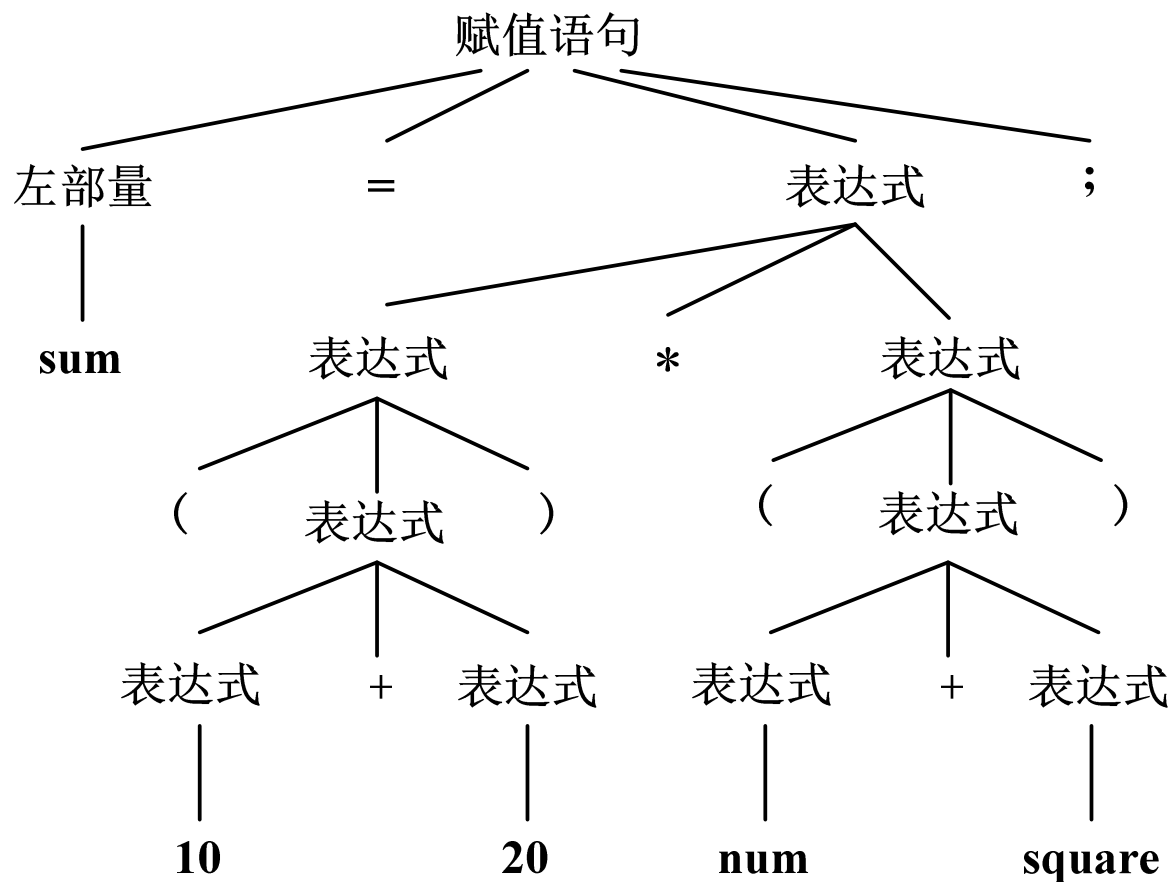
- **语法分析由语法分析器(Syntax Analyzer)完成，语法分析器又叫Parser。**
- **功能：**
 - **Parser实现“组词成句”：将词组成各类语法成分，例如因子、项、表达式、语句、子程序...**
 - **构造分析树**
 - **指出语法错误**
 - **指导翻译**
- **输入：token序列**
- **输出：语法成分**

2、语法分析

`sum=(10+20)*(num+square);`

词法分析结果

- (标识符, sum)
- (赋值号, =)
- (左括号, (
- (整常数, 10)
- (加号, +)
- (整常数, 20)
- (右括号,)
- (乘号, *)
- (左括号, (
- (标识符, num)
- (加号, +)
- (标识符, square)
- (右括号,)
- (分号, ;)



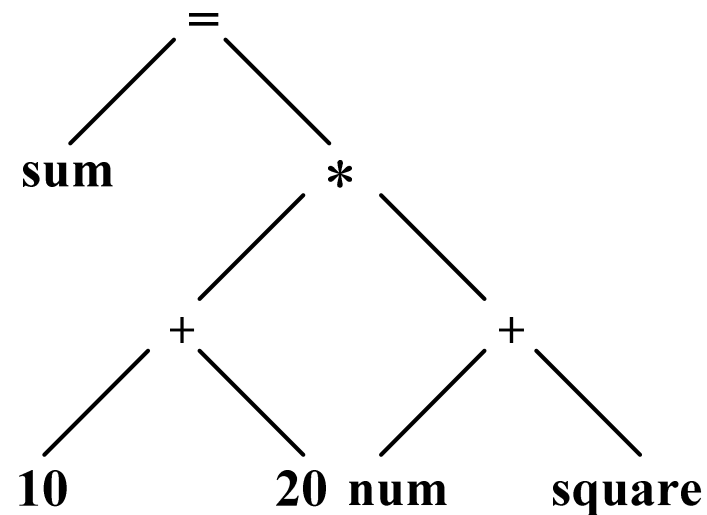
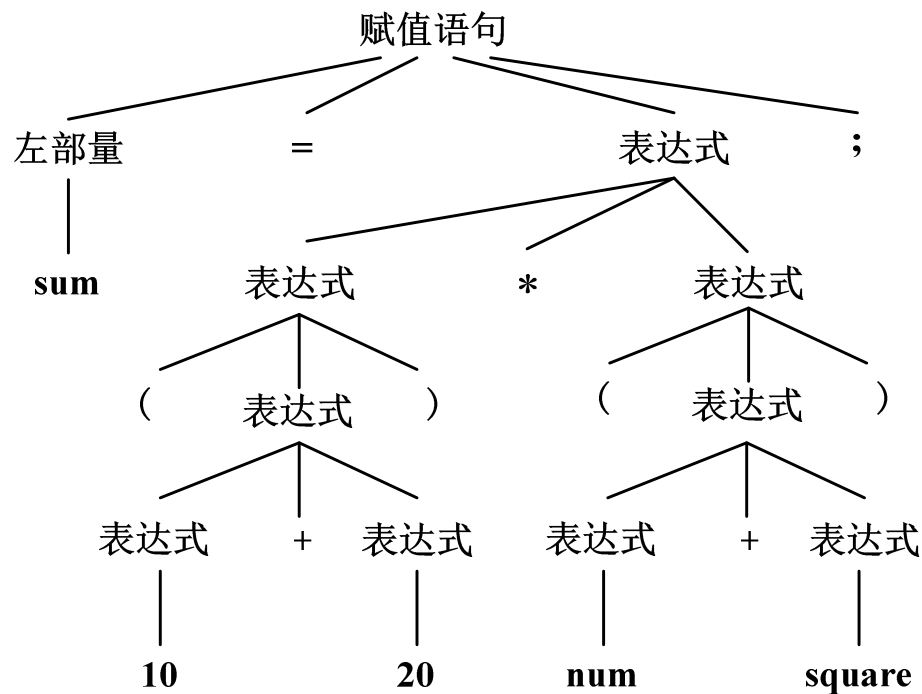


3、语义分析

- 语义分析(semantic analysis)一般和语法分析同时进行，称为**语法制导翻译**(syntax-directed translation)
- 功能：分析由语法分析器识别出来的语法成分的语义
 - 获取标识符的属性：类型、作用域等
 - 语义检查：运算的合法性、取值范围等
 - 子程序的静态绑定：代码的相对地址
 - 变量的静态绑定：数据的相对地址

3、语义分析

sum=(10+20)*(num+square);



4、中间代码生成

- 语义分析通常以中间代码形式表达操作
- 例: $\text{sum} = (10 + 20) * (\text{num} + \text{square});$

后缀表示(逆波兰Anti- Polish Notation)

$\text{sum } 10 \ 20 \ + \ \text{num } \text{square} \ + \ * =$

前缀表示(波兰Polish Notation)

$= \text{sum } * + 10 \ 20 + \text{num } \text{square}$

四元式表示

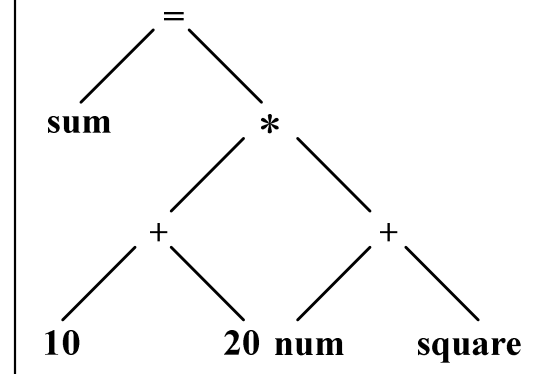
(三地址码)

$(+, 10, 20, T_1)$
 $(+, \text{num}, \text{square}, T_2)$
 $(*, T_1, T_2, T_3)$
 $(=, T_3, , \text{sum})$

三元式表示

$(+, 10, 20)$
 $(+, \text{num}, \text{square})$
 $(*, (1), (2))$
 $(=, \text{sum}, (3))$

树表示





4、中间代码生成

- **中间代码的特点**
 - **简单规范**
 - **与机器无关**
 - **易于优化与转换**



5、代码优化

■ 先谈谈代码优化的一些问题

- Donald Knuth: 过早的优化是万恶之源，让正确的程序更快，要比让快速的程序正确容易得多
- 将最多的努力投入到**运行消耗时间最多**的那部分代码中
- 优化的层次越高，就会越有效，最好的优化是找到一个更有效的算法
- 将程序编写和优化当作独立的步骤来做
- 性能分析数据应该是第一位，最后才是直觉



5、代码优化

- **代码优化(optimization)**
 - **对中间代码进行优化处理，使程序运行能够尽量节省存储空间，更有效地利用机器资源，使得程序的运行速度更快，效率更高**
 - **这种优化变换必须是等价的。**
 - **分为（与机器无关的优化）和（与机器有关的优化）两种**



与机器无关的优化

■ 局部优化

- 常量合并：常数运算在编译期间完成，如 $8+9*4$
- 公共子表达式的提取

■ 全局优化

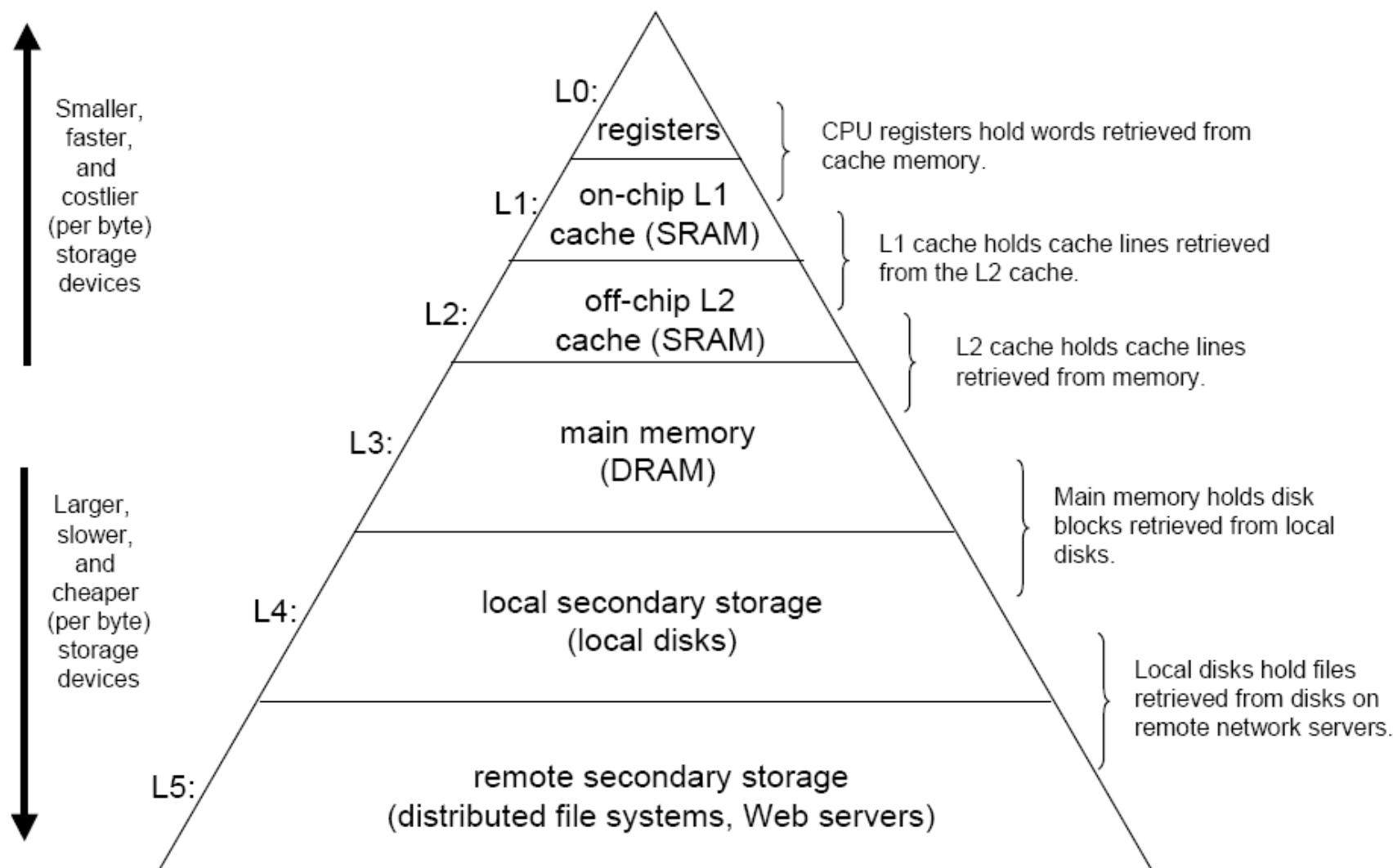
- 主要是指循环优化
- 强度削减：用较快的操作代替较慢的操作
- 代码外提：将循环中不变的计算移出循环



与机器有关的优化

- **寄存器的利用**
 - **将常用量放入寄存器，以减少访问内存的次数**
- **体系结构**
 - **SIMD、MIMD、SPMD、向量机、流水机**
- **任务划分**
 - **按运行的算法及体系结构，划分子任务(MPMD)**
- **存储策略**
 - **根据算法访存的要求安排：Cache、并行存储体系——减少访问冲突**

与机器有关的优化-存储层次





与机器有关的优化-存储层次

■ CPU寄存器

- 保存着最常用的数据，零个周期访问数据

■ 高速缓冲存储器

- 靠近CPU的小的、快速的存储器，1-30个周期访问数据

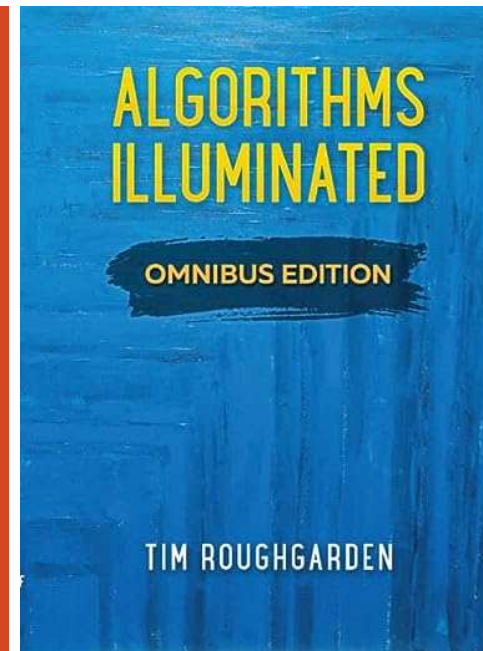
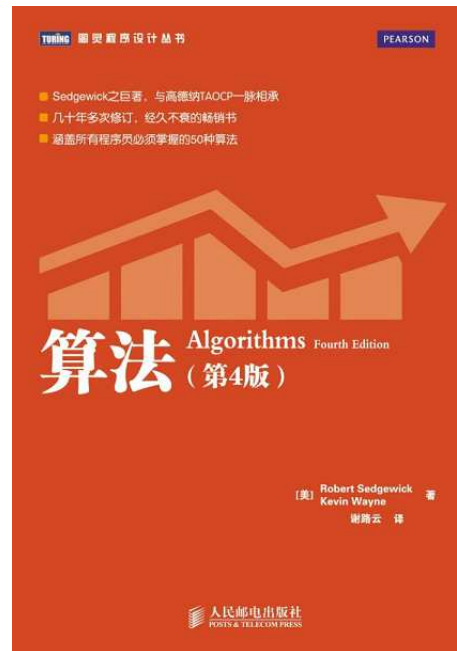
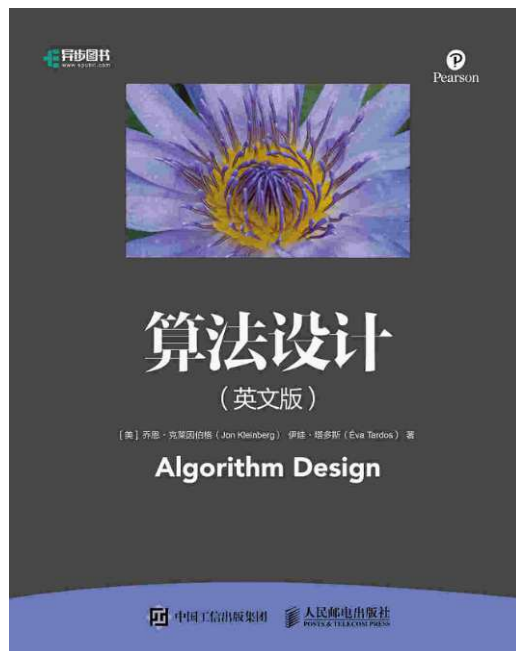
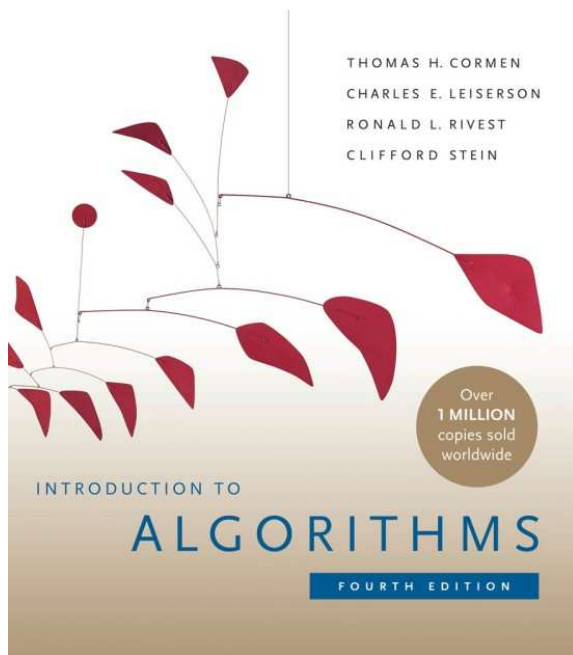
■ 主存储器

- 存储系统和进程运行所需的数据和指令，50-200个周期访问数据

■ 磁盘

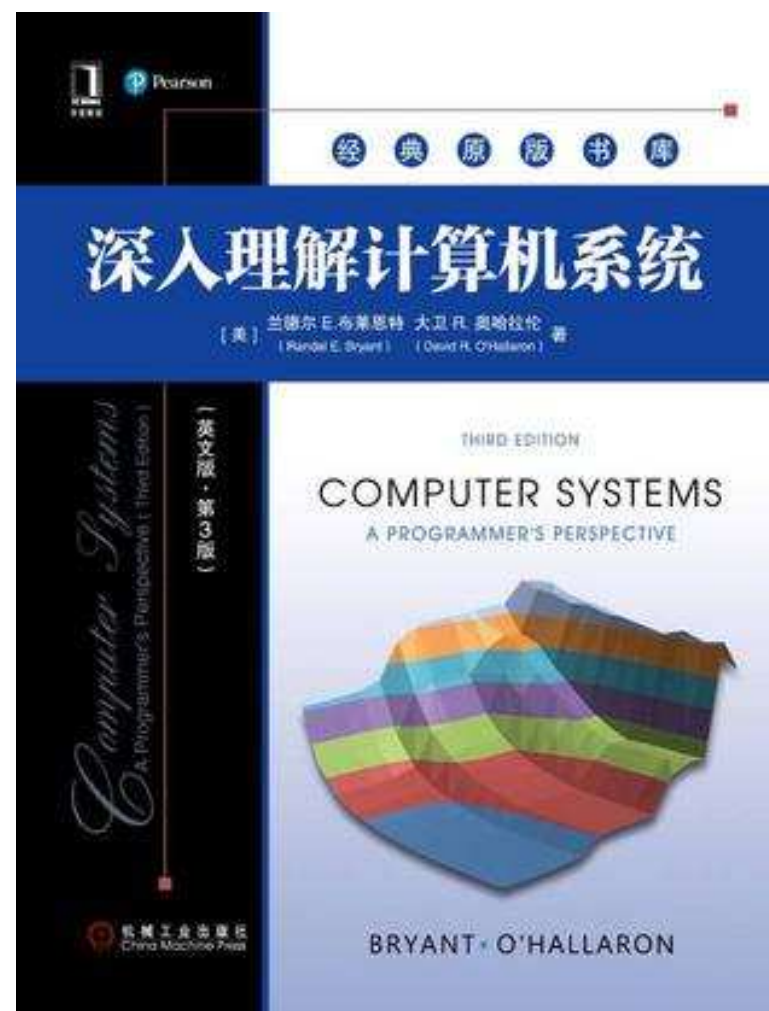
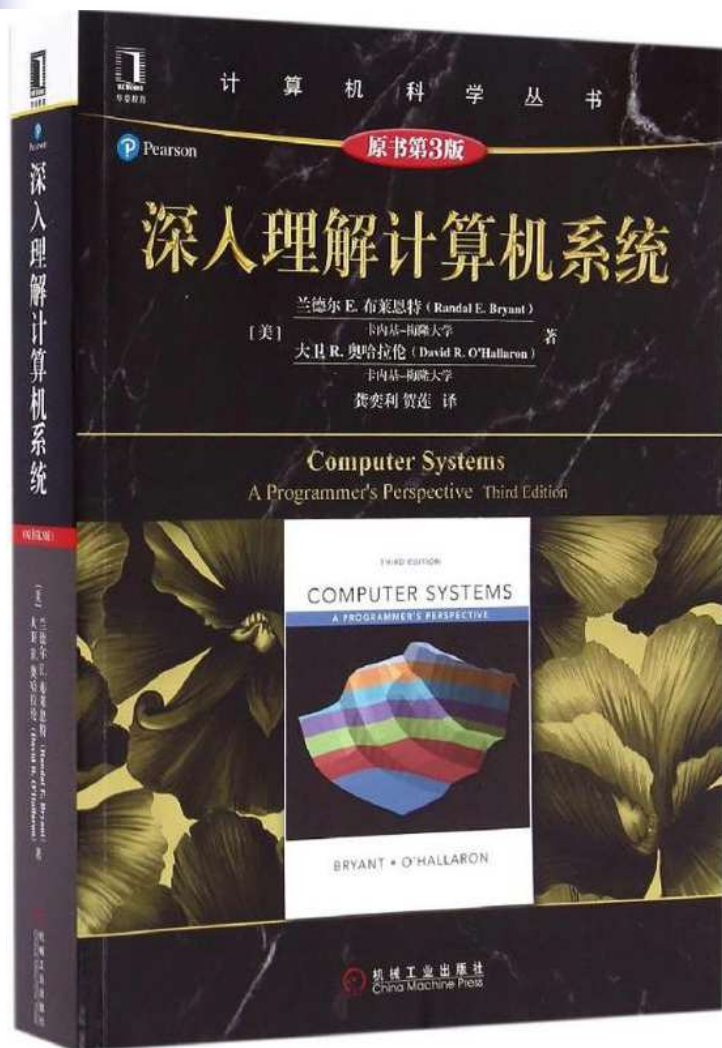
- 最主要存储设备，10,000,000个周期访问数据

与机器无关的优化





与机器有关的优化





6、目标代码生成

- **编译程序的最后一个阶段**
- **为中间代码中出现的运算对象分配存储单元、寄存器等**
- **将中间代码转换成目标机上的机器指令代码或汇编代码**
 - **对于源语言的各种语法成分，确定目标代码结构（机器指令组/汇编语句组）**
 - **制定从中间代码到目标代码的翻译策略或算法**

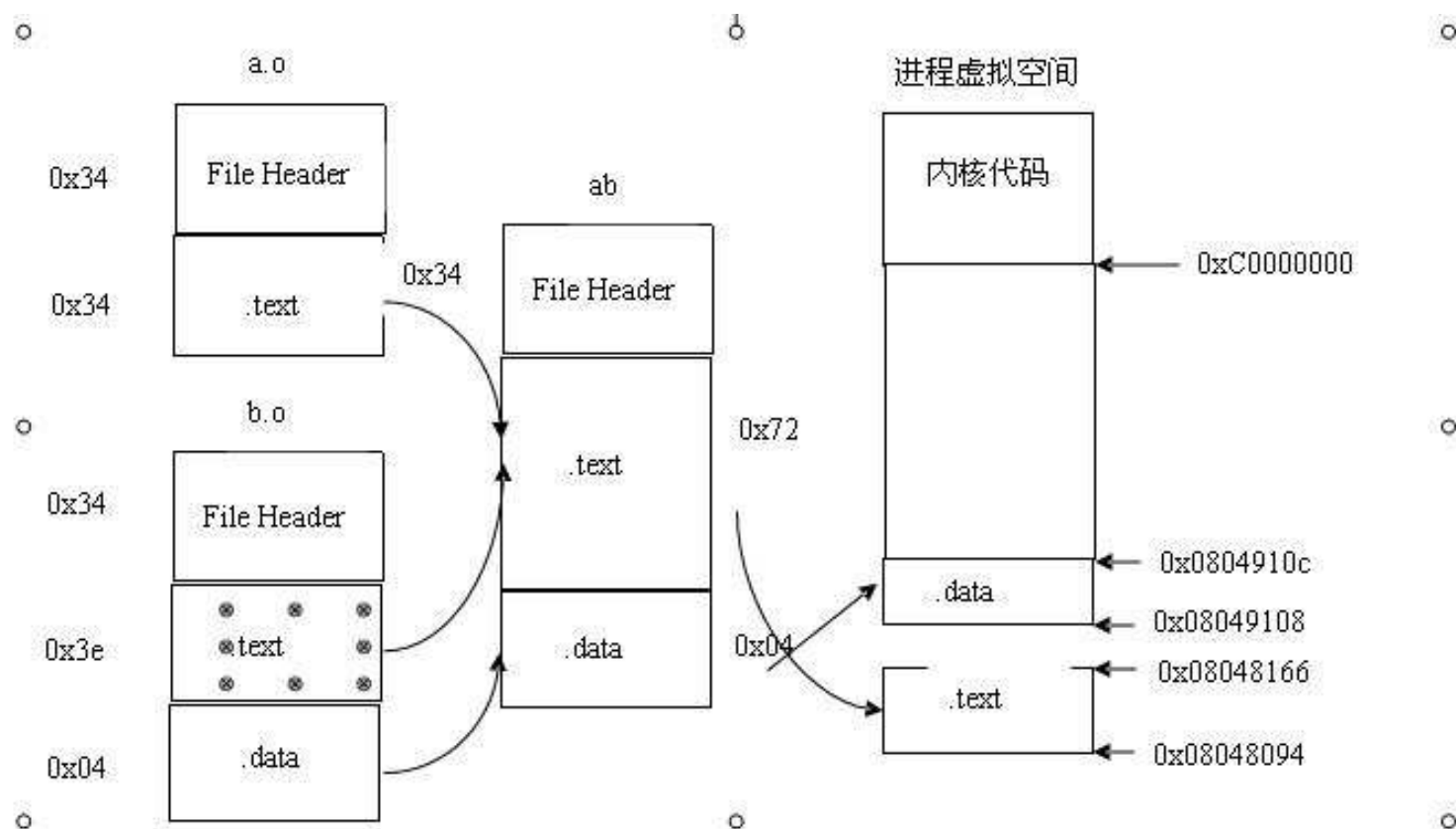


6、目标代码生成

- **目标代码的形式**
 - **具有绝对地址的机器指令**
 - **汇编语言形式的目标程序**
 - **模块结构的机器指令（需要链接程序）**

6、目标代码生成

■ 目标代码的形式





7、错误处理

- **进行各种错误的检查、报告、纠正，以及相应的续编译处理(如：错误的定位与局部化)**
 - **词法分析阶段：拼写方面的错误，出现非法字符等**
 - **语法分析阶段：表达式、句子或程序结构等错误**
 - **语义分析阶段：类型匹配错误、参数匹配错误、非法转移问题等**



8、表格管理

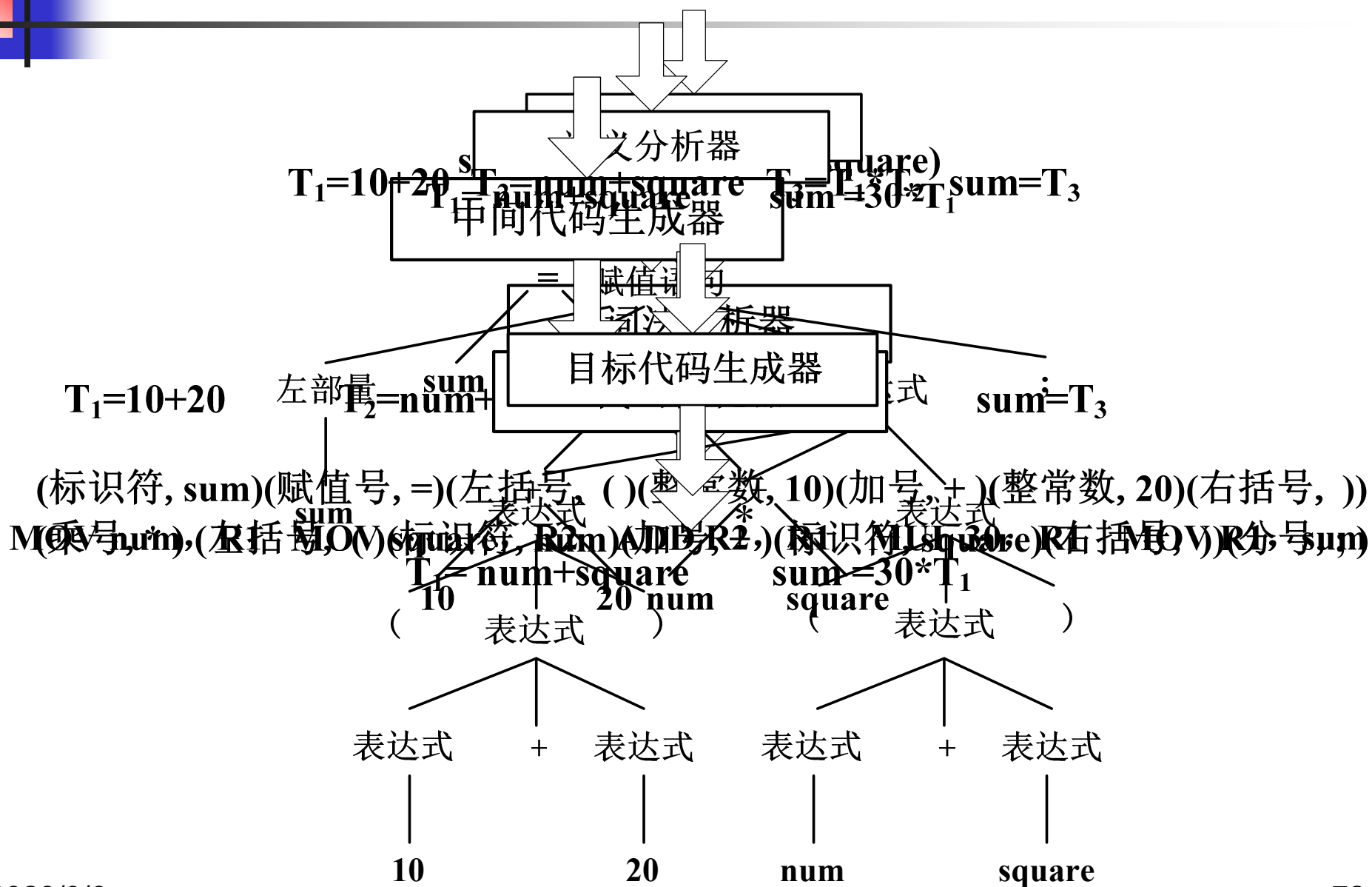
- **管理各种符号表(常数、标号、变量、过程、结构.....)，查、填源程序中出现的符号和编译程序生成的符号，为编译的各个阶段提供信息。**
- **Hash表、链表等各种表的查、填技术**
- **“数据结构与算法”课程的应用**



模块分类

- **分析**：词法分析、语法分析、语义分析
- **综合**：中间代码生成、代码优化、目标代码生成
- **辅助**：符号表管理、出错处理
- **8项功能对应8个模块**

语句 $\text{sum}=(10+20)*(\text{num}+\text{square});$ 的翻译过程



1.4 编译程序的组织

- 编译程序的翻译分成词法分析、语法分析、语义分析与中间代码生成、代码优化、目标代码生成五个阶段
- 根据系统资源的状况、运行目标的要求等，可以将一个编译程序设计成多遍 (Pass) 扫描的形式，在每一遍扫描中，完成不同任务。
 - 如：首遍构造语法树，二遍处理中间表示，增加信息等。

通常称编译程序对源程序或中间结果的完整扫描为**遍**



1.4 编译程序的组织

- 遍可以和阶段相对应，也可以和阶段无关
- 遍数量的优化
 - 根据语言、系统追求的目标、计算机的资源状况等决定

1.4 编译程序的组织

■ 多遍扫描

- 本遍扫描的结果作为下一遍扫描的输入，本遍扫描中得到的信息在下一遍扫描中更容易获得更优化的程序。
- 增加内存访问次数，可

■ 单遍扫描

- 分析所需的信息可能因单遍扫描而难以达到最优

一般来说，可以将
词法分析、语法分析、
语义分析和中间代码生
成做成一遍
将代码优化做成一遍
将目标代码生成做成一
遍



1.4 编译程序的组织

- **编译程序的设计目标**
 - **规模小、速度快、诊断能力强、可靠性高、可移植性好、可扩充性好**
 - **目标程序也要规模小、执行速度快**
- **编译系统规模较大，因此可移植性很重要**
 - **为了提高可移植性，将编译程序划分为前端和后端**



1.4 编译程序的组织

■ 前端

■ 与源语言有关、与目标机无关的部分

对于某一种高级语言在不同机器上的编译系统来说，前端的处理基本是一样的，前端部分可被复用，只需要针对不同的机器构建后端就可以

分析、译码、代码生成

部分
优化、

在某一种机器上实现多种高级语言的编译系统，后端部分可以被复用，只需要针对不同高级语言构建前端就可以



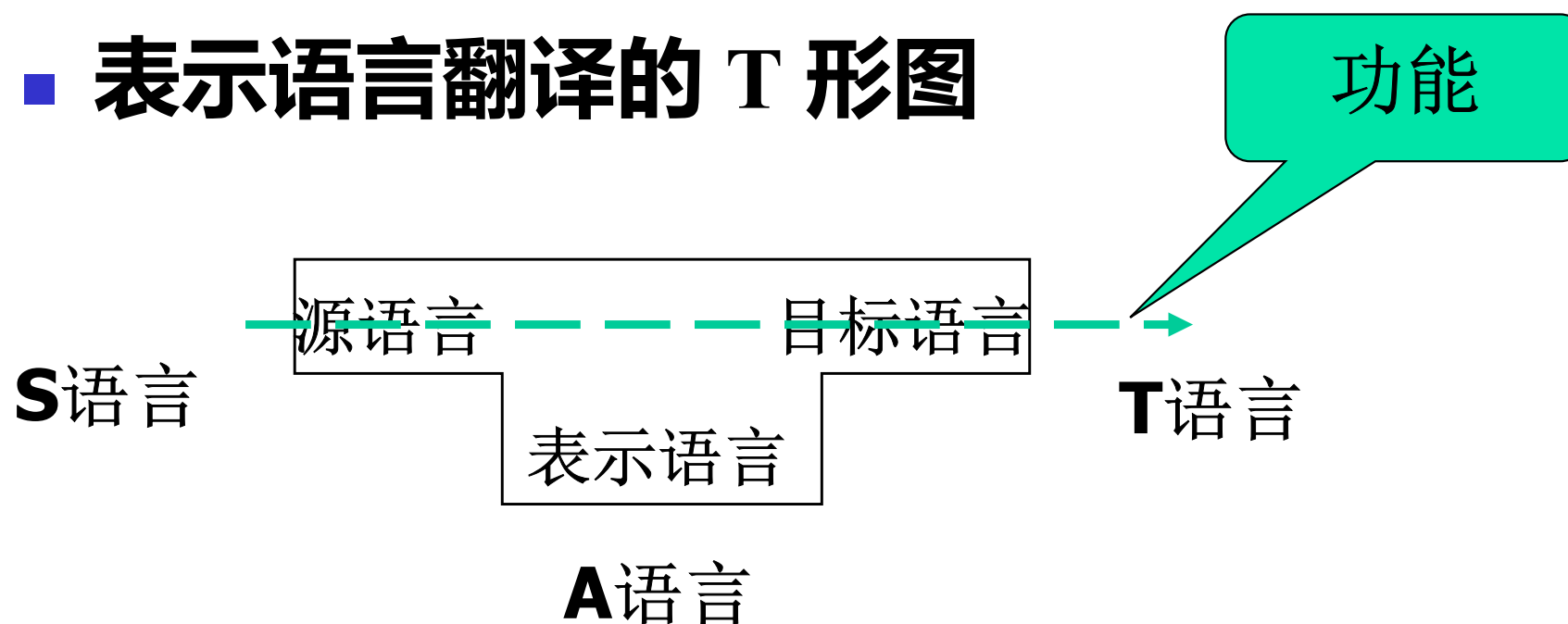
1.5 编译程序的生成

■ 如何实现编译器？

- 编译程序也是运行在计算机上的程序
- 1970年前，机器代码编写
- 1980年后，高级语言编写
- 利用现有的工具和编译程序，通过移植、自展等方法完成新的编译程序

1. T形图

■ 表示语言翻译的 T 形图



一个用**A**语言描述的编译程序，它将**S**语言源程序翻译成了**T**语言目标程序



2. 自展

- 问题一：如何直接在一个机器上实现C语言编译器？
- 解决：
 - 用汇编语言实现一个C子集的编译程序 P_0
 - 用汇编程序处理该程序 P_0 , 得到 P_2 （可直接运行）
 - 用C子集编制C语言的编译程序 P_3
 - 用 P_2 编译 P_3 , 得到 P_4



3.移植

- **问题二：A机上有一个C语言编译器，是否可利用此编译器实现B机上的C语言编译器？**

- **条件：A机有C 语言的编译程序**
- **目的：实现B机的C语言的编译**

解决办法

- 1. 用 C 语言编制B机的 C 编译程序 P_0**
- 2.(A 机C编译器 P_1)编译 P_0 ，得到在A机可运行的 P_2**
- 3.(A 机的 P_2)编译 P_0 ，得到B机上可运行的编译器 P_3**



4.本机编译器的利用

- **问题三： A机上有一个C语言编译器， 现要实现一个新语言NEW的编译器？**

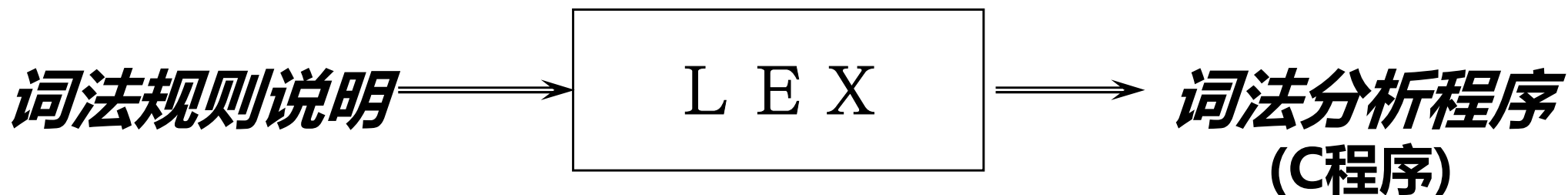
解决方法：

用**C**编写**NEW**的编译器， 并用**C**编译器编译它



5. 编译程序的自动生成

1) 词法分析器的自动生成程序



输入:

词法 (正规表达式)
识别动作 (C 程序段)

输出:

yylex() 函数

2) 语法分析器的自动生成程序



输入：

语法规则 (产生式)
语义动作 (C程序段)

输出：

yyparse() 函数



1.6 本章小结

- **编译原理是一门非常好的课**
- **程序设计语言及其发展**
- **程序设计语言的翻译**
- **编译程序的总体结构**
- **编译程序的各个阶段**
- **编译程序的组织与生成**