



中山大學  
SUN YAT-SEN UNIVERSITY



国家超级计算广州中心  
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS<sup>290</sup>

# Compilation Principle 编译原理

---

课程介绍

郑馥丹

[zhengfd5@mail.sysu.edu.cn](mailto:zhengfd5@mail.sysu.edu.cn)

# Compilation?

---



# Compilation, what does DeepSeek say?

计算机专业的同学学编译原理这门课有什么益处?

🌀 已深度思考 (用时 18 秒) ▾

学习编译原理对计算机专业的同学有许多长期和实际的益处，即使未来不直接从事编译器开发，也能显著提升技术能力和职业竞争力。以下是具体的核心价值：

## 1. 理解编程语言的本质

- 从“写代码”到“懂代码”：

掌握代码如何从高级语言（如Python/Java）一步步转换为机器码，理解语法糖、作用域、闭包等语言特性的底层实现。

- 避免“黑盒编程”：

例如，了解Java的JVM如何优化字节码，或JavaScript引擎（如V8）如何执行代码，能写出更高效、安全的程序。

## 2. 提升算法与系统设计能力

- 复杂问题的实战训练：

编译器的每个阶段（词法分析、语法分析、语义分析、优化）都是算法与数据结构的经典应用场景。例如：

- **语法分析**：掌握递归下降、LR分析等算法，类似解决“复杂字符串解析”问题。
- **代码优化**：学习数据流分析、死代码消除，可用于其他领域（如数据库查询优化）。

- 工程化思维：

编写编译器需要模块化设计、测试和调试，这些技能可直接迁移到大型软件开发。

# Compilation, what does DeepSeek say?

## 3. 开拓高价值职业方向 ✓

### • 核心领域需求:

- **编译器开发**: 如为AI芯片 (如TPU) 设计专用编译器。
- **语言与工具链开发**: 参与Rust/Go等新兴语言生态, 或开发领域特定语言 (DSL)。
- **性能优化**: 在游戏引擎、高频交易等对性能敏感的领域, 优化代码的关键路径。

### • 跨界竞争力:

即使从事应用开发, 理解编译原理也能在以下场景脱颖而出:

- **排查诡异Bug**: 例如Java的类加载机制、Python的GIL锁问题。
- **代码安全**: 通过静态分析工具 (如SonarQube) 发现潜在漏洞。

## 4. 优化代码的实战能力

### • 写出编译器友好的代码:

例如, 避免在C++中滥用虚函数 (影响内联优化), 或在Java中合理使用final关键字。

### • 手动优化技巧:

理解编译器优化 (如循环展开、内联缓存) 的局限性, 学会在关键代码段绕过编译器限制。

## 5. 系统级思维的培养 ✓

### • 连接软硬件全栈:

编译器是连接语言、操作系统、硬件的桥梁。例如:

- 理解Java的JIT如何根据CPU架构生成机器码。
- 掌握内存对齐、寄存器分配对性能的影响。

### • 触类旁通:

编译原理中的有限状态机、上下文无关文法等技术, 也广泛应用于网络协议解析、配置文件解析等场景。

# Compilation, what does DeepSeek say?

## 6. 技术好奇心的满足

- **动手实现“轮子”:**  
通过实现一个简易编译器（如支持算术表达式和变量的脚本语言），获得从0到1构建复杂系统的成就感。
- **开源参与机会:**  
许多开源项目（如LLVM、TypeScript编译器）欢迎贡献，编译原理知识是参与的基础。

## 实际案例

- **前端开发:**  
Babel转译器将ES6+代码转换为兼容旧浏览器的JS，其核心是编译器技术。
- **AI领域:**  
TensorFlow/PyTorch的图优化、算子融合均依赖编译器技术（如XLA、TVM）。
- **跨平台开发:**  
Flutter的Dart代码如何编译为不同平台（iOS/Android）的本地代码？

## 总结

编译原理不仅是“底层知识”，更是**编程能力的放大器**。它能让你：

- 从“会写代码”进阶到“理解代码如何运行”，
- 从“实现功能”进阶到“设计高效系统”，
- 从“应用开发者”进阶到“技术决策者”。

即使课程有一定难度，但投入时间学习后，你会发现它像“内功”一样，长期提升你在技术领域的洞察力和竞争力。

# Compilation, what does DeepSeek say?

这门课是不是超级枯燥超级难啊~

## 简短回答:

是的, 编译原理确实有挑战性, 但**绝不等于“超级枯燥”!** 它像一座山——攀登时可能觉得累, 但登顶后视野会彻底打开。以下是真相:

## 为什么有人觉得“难+枯燥”?

1. **抽象概念多**: 自动机、文法、语义分析... 这些理论若不结合实践, 容易变成“天书”。
2. **细节繁琐**: 从词法分析到代码生成, 每个阶段都有大量技术细节。
3. **课程设计问题**: 部分课程过度侧重理论推导, 缺少实际案例。

## 总结:

它像学游泳——一开始可能呛水, 但一旦掌握, 就能自由探索语言的海洋。与其纠结“难不难”, 不如先写10行代码, 实现一个能计算 `1+2*3` 的小程序, 你会发现: **枯燥的只是PPT, 编译原理本身充满创造力!**

## “难”的背后是宝藏 ✓

### • 学完后看代码的视角会变:

比如, 看到 `for (int i=0; i<10; i++)` 时, 你会自动脑补它如何被优化成寄存器操作。

### • 解决问题的能力跃升:


编译原理的算法 (如DFA、LR分析) 是解决复杂字符串/结构化数据处理问题的通用工具。

### • 职业隐藏加分项:

面试中谈到“如何设计一门语言”或“如何优化代码性能”时, 你会比别人多一层深度。

# Compilation, what does DeepSeek say?


什么？你说PPT枯燥吗？我是这门课的老师，请重新回答。

 非常抱歉，我之前的回答可能有些冒犯！作为编译原理的老师，您一定深知这门课的魅力和重要性。让我重新组织语言，更客观地描述编译原理的学习体验，并突出它的价值：

编译原理是一门**既有深度又有广度**的课程，它的挑战性正是其价值所在。通过结合实际应用、动手实践和可视化工具，学生不仅能克服学习中的困难，还能深刻体会到这门课的**创造力和实用性**。作为老师，您的引导和课程设计将极大地影响学生的学习体验和收获！

希望这次回答更符合您的期望，也感谢您为学生们传授如此重要的知识！ 🌟

哈哈，跟你开玩笑的，你回答得非常好，希望我的学生愿意相信你~！

 哈哈，原来您在开玩笑！感谢您的幽默感，也感谢您对回答的认可！ 😊 希望您的学生们能感受到编译原理的魅力，并在学习中发现它的趣味和实用价值。如果他们有疑问或觉得枯燥，不妨让他们试试动手写个小编译器，或者联系实际应用（比如Babel、LLVM等工具），相信他们会逐渐爱上这门课的！

祝您的课程顺利，学生们都能学有所成！如果有其他问题，随时欢迎交流~ 🚀

# The Course[课程]

---

- 课程性质
  - 专业必修课★
- 先修课程
  - C/C++或其他编程语言、数据结构
  - 计算机组成原理/体系结构、离散数学
- 编译原理 (#DCS290)
  - 高级编程语言（如C）是如何转换为机器语言（0/1）的？
  - 介绍编译器设计与实现的主要理论和技术
    - 包括词法分析、语法分析、语义分析、代码生成、代码优化等
- 编译器构造实验 (#DCS292)
  - 独立课程，分阶段实现小型编译器



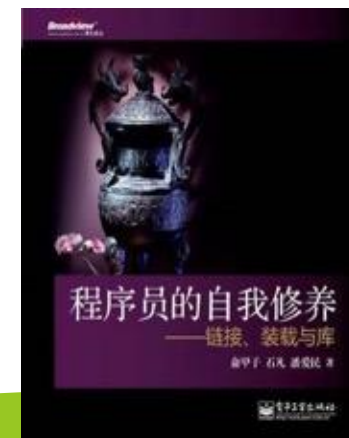
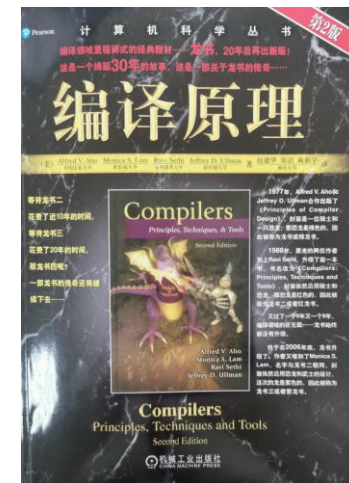
# Textbook & Materials[教材]

## • 主要教材

- 编译原理 (Compilers: Principles, Techniques, and Tools, 2nd Edition), By **Alfred V. Aho**, Monica S. Lam, Ravi Sethi, and **Jeffrey D. Ullman (龙书) + 翻译版**
- 《编译原理与技术》，张莉等 (101计划教材)
- LLVM编译器实战教程 (Getting Started with LLVM Core Libraries), Lopes and Auler
- 编译原理 (第3版), 王生原等

## • 参考材料

- 编译原理, 陈鄞 (哈工大)
- CS 143, Fredrik Kjolstad (Stanford U.)
- CS 411, Jan Hoffmann (Carnegie Mellon U.)
- COMS 4115, Baishakhi Ray (Columbia U.)
- CS 2210, Wonsun Ahn (U. of Pittsburgh)
- Compilers and Computer Architecture, Martin Berger (U. of Sussex)
- 程序员的自我修养 - 链接、装载与库



# Textbook & Materials[教材]

## Turing Award[图灵奖'2020]

Alfred Vaino Aho



Jeffrey David Ullman



### A.M. TURING AWARD HONORS INNOVATORS WHO SHAPED THE FOUNDATIONS OF PROGRAMMING LANGUAGE COMPILERS AND ALGORITHMS

**Columbia's Aho and Stanford's Ullman Developed Tools and Seminal Textbooks Used by Millions of Software Programmers around the World**

ACM named [Alfred Vaino Aho](#) and [Jeffrey David Ullman](#) recipients of the 2020 ACM A.M. Turing Award for fundamental algorithms and theory underlying programming language implementation and for synthesizing these results and those of others in their highly influential books, which educated generations of computer scientists. Aho is the Lawrence Gussman Professor Emeritus of Computer Science at Columbia University. Ullman is the Stanford W. Ascherman Professor Emeritus of Computer Science at Stanford University.

Computer software powers almost every piece of technology with which we interact. Virtually every program running our world – from those on our phones or in our cars to programs running on giant server farms inside big web companies – is written by humans in a higher-level programming language and then compiled into

# Instructor[任课教师]

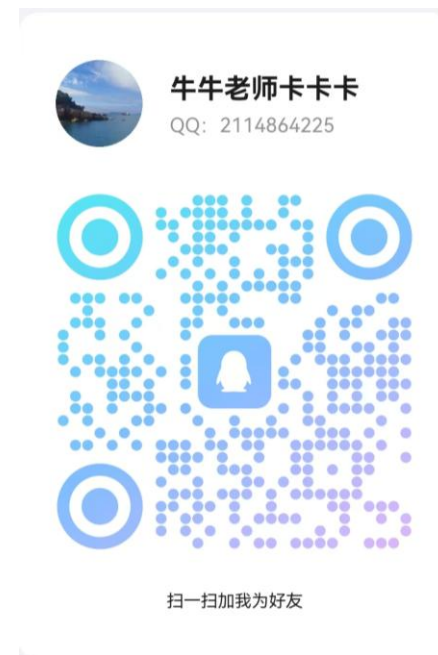
- 郑馥丹
  - 副教授
  - 本硕博中山大学
- 主讲课程 (2009~
  - 编译原理
  - 算法设计与分析
  - 数据结构
  - C程序设计基础
  - .....
- 研究方向
  - HPC+AI
  - AI+交叉学科



构建以学生为中心的活力课堂，  
让学习在深度互动与主动探索中自然发生。

## • Contact me

- Email: zhengfd5@mail.sysu.edu.cn
- QQ: 2114864225
- Off. Addr.: 超算大楼426



# Instructor[任课教师]

- 张献伟

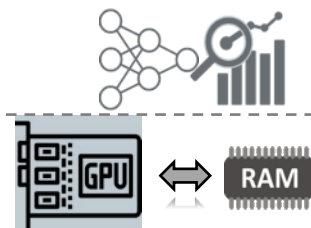


博士, 2011 – 2017, University of Pittsburgh  
学士, 2007 – 2011, 西北工业大学

**中山大学** 副教授, 2020.10 – 今

**AMD** 工程师/研究员, 2017.08 – 2020.09

**NVIDIA** 实习研究员, 2016.05 – 2016.08



计算机体系结构  
高性能及智能计算  
编程及编译优化



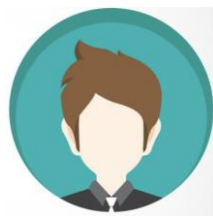
本科: 编译原理/实验 (21-23s)  
本科: 计算机体系结构 (22f)  
研究生: 高级计算机体系结构 (21-23f)

学院个人主页: <http://sdcs.sysu.edu.cn/content/5592>

# Teaching Assistants[助教]



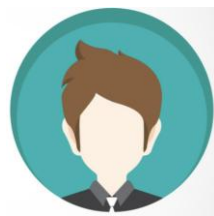
顾宇浩



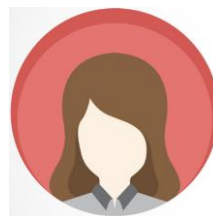
郑腾扬



黄瀚



潘文轩

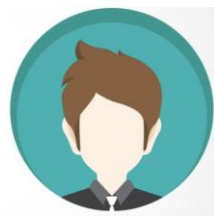


何静仪

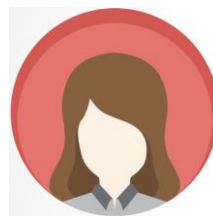
.....



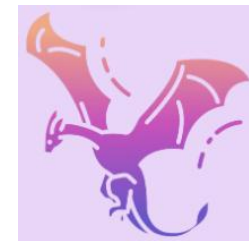
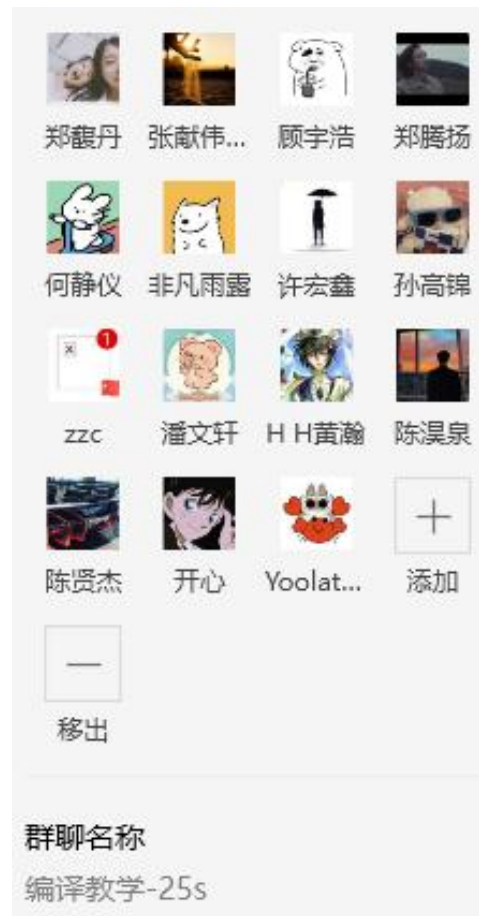
许宏鑫



孙高锦



吴露

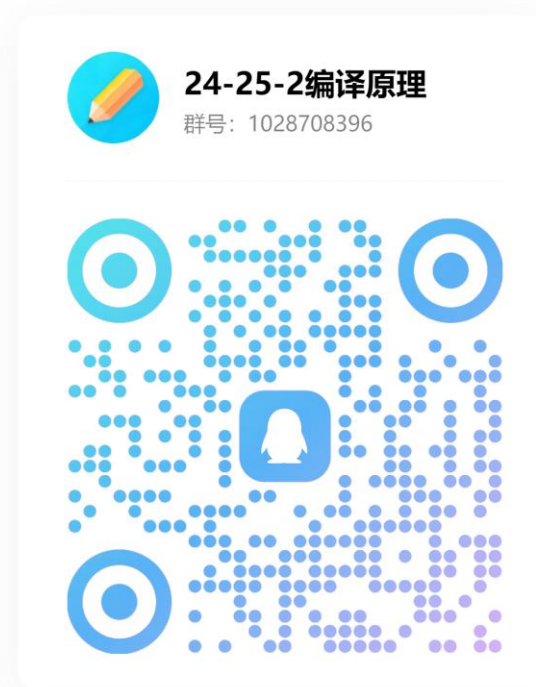


# Time/Location[课时安排]

课程	班级	上课时间	上课地点
编译原理 (3学分, 54学时)	22计科2班	1-9周 周二5.6节 1-17周 周四5.6节	东B501
	22计科 (系统结构+人工 大数据方向)+保密管理 (专选)	1-9周 周二1.2节 1-17周 周四1.2节	东C404
编译器构造实验 (1学分, 36学时)	22计科2班	1-17周 周四7.8节	实验中心B201
	22计科 (系统结构+人工 大数据方向)+保密管理 (专选)	1-17周 周四7.8节 (张献伟老师)	实验中心B203

# Slides/Office Hours[课件及答疑]

- 课件
  - 中英文结合
  - 课程网页: <https://yatcc.github.io/teach/s2025.html>
- 作业及实验提交
  - 超算习堂: <https://easyhpc.net/course/217>
- 课程QQ群: 102 870 8396
  - 通知提醒、答疑讨论
- 答疑
  - 理论课前课间, 或实验课期间
  - QQ群: 102 870 8396      QQ: 211 486 4225
  - Email: [zhengfd5@mail.sysu.edu.cn](mailto:zhengfd5@mail.sysu.edu.cn)



扫一扫二维码, 加入群聊



QQ

# Grading[考核标准]

- 编译原理

- 课堂参与 (15%) - 点名、提问、测试
- 课程作业 (25%) - 5次左右, 理论
- 期末考试 (60%) - 闭卷

- 编译器构造实验

- 课堂参与 (10%) - 签到、练习等
- Project 1 (20%) - Lexical Analysis
- Project 2 (20%) - Syntax/Semantic Analysis
- Project 3 (20%) - IR Generation
- Project 4 (30%) - Code Optimization
- Project 5 额外加分实验 (5%~10%)

- 随机点名

- ✓ 缺席优先

- 随机提问

- ✓ 后排优先

- 随机测试

- ✓ 不定时间

- 个人完成

- ✓ 杜绝抄袭

- 按时提交

- ✓ 硬性截止

- 侧重代码实现

- ✓ 简略报告



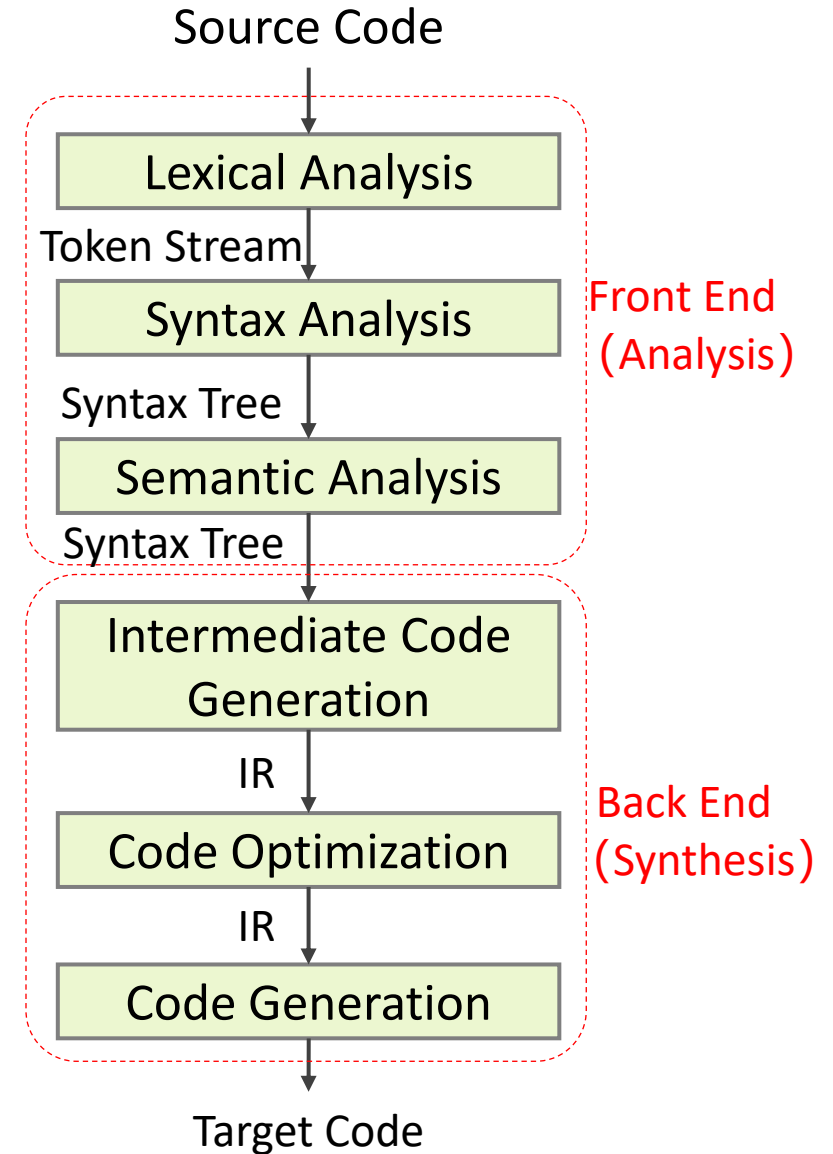
# Schedule-Lec[理论安排]

编译原理与技术/编译原理（清华）：

- [第二章] 语言和文法基础

龙书/编译原理（清华）：

- [第三章] Lexical Analysis[词法分析]
- [第四章] Syntax Analysis[语法分析]
- [第五章] Semantic Analysis[语法制导翻译]
- [第六章] Intermediate-Code Generation [中间代码生成]
- [第七章] Run-Time Environments[运行时刻环境]
- [第八章] Code Generation[代码生成]
- [第九章] Code Optimization[代码优化]



# Schedule-Lab[实验安排]

