



第1讲

编译原理和技术导论

李 诚、徐伟

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

先进技术研究院、计算机科学与技术学院

2025年02月27日

1

编译器是什么？

2

为什么要学习编译课程？

3

编译教学的困境与科大方案

4

本学期课程设置的情况

什么是/为什么需要编译器？



□ 高级语言

- 直接面向开发者
- 与数学公式类似
- 编程效率高

□ 机器语言

- 驱动硬件完成具体任务
- 编程效率低

□ 编译器提供程序开发的便捷性

- 实现人机交流，将人类易懂的高级语言翻译成硬件可执行的目标机器语言

从C程序到可执行文件



```
#include <stdio.h>
int main()
{
    printf("hello, world!\n");
}

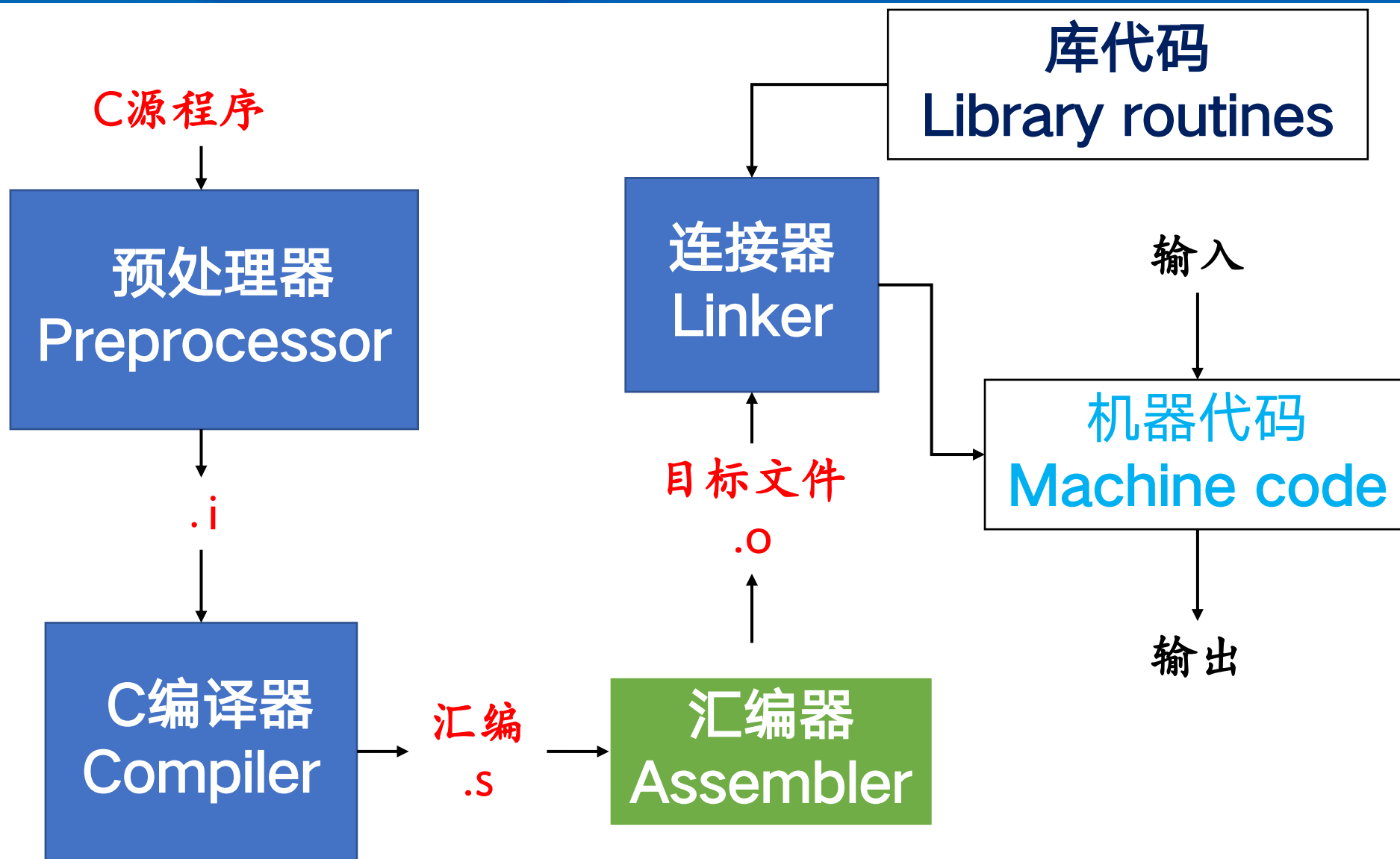
/* helloworld.c */
```

```
[root@host ~]# clang helloworld.c -o helloworld
```

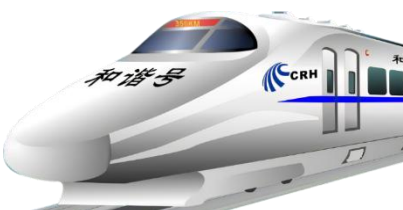
```
[root@host ~]# ./helloworld
```

```
hello, world!
```

C程序的编译运行过程分解



编译器自动优化能力



优化等级	简要说明
-Ofast	在-O3级别的基础上，开启更多 激进优化项 ，该优化等级不会严格遵循语言标准
-O3	在-O2级别的基础上，开启了更多的 高级优化项 ，以编译时间、代码大小、内存为代价获取更高的性能。
-Os	在-O2级别的基础上，开启 降低生成代码体量 的优化
-O2	开启了大多数 中级优化 ，会改善编译时间开销和最终生成代码性能
-O/-O1	优化效果介于-O0和-O2之间
-O0	默认优化等级，即 不开启编译优化 ，只尝试减少编译时间

延伸阅读：<https://clang.llvm.org/docs/CommandGuide/clang.html#code-generation-options>

举例——优化对代码性能的影响



❑ 1 0000000000次循环迭代累加

```
#include <stdio.h>
#include <time.h>
int main() {
    int loop = 1000000000;
    long sum = 0;
    int start_time = clock();
    int index = 0;
    for (index = 0; index < loop; index
        {
            sum += index;
        }
    int end_time = clock();
    printf("Sum : %ld, Time Cost : %lf \n", sum, (end_time - start_time) * 1.0 /
CLOCKS_PER_SEC);
    return 0;
}
```

循环次数定义

开始计时

循环体

结束计时

代码运行时间输出

举例——优化对代码性能的影响



□gcc -O0 无优化执行

```
gloit@gloit-x1c > ~/2022_compiler_demo } master gcc -O0 add.c
gloit@gloit-x1c > ~/2022_compiler_demo } master ./a.out
Sum: 499999999500000000, Time Cost: 3.415244
```

□gcc -O1 中级优化执行

```
gloit@gloit-x1c > ~/2022_compiler_demo } master gcc -O1 add.c
gloit@gloit-x1c > ~/2022_compiler_demo } master ./a.out
Sum: 499999999500000000, Time Cost: 0.554717
```

性能提升5倍

□gcc -O2 高级优化执行

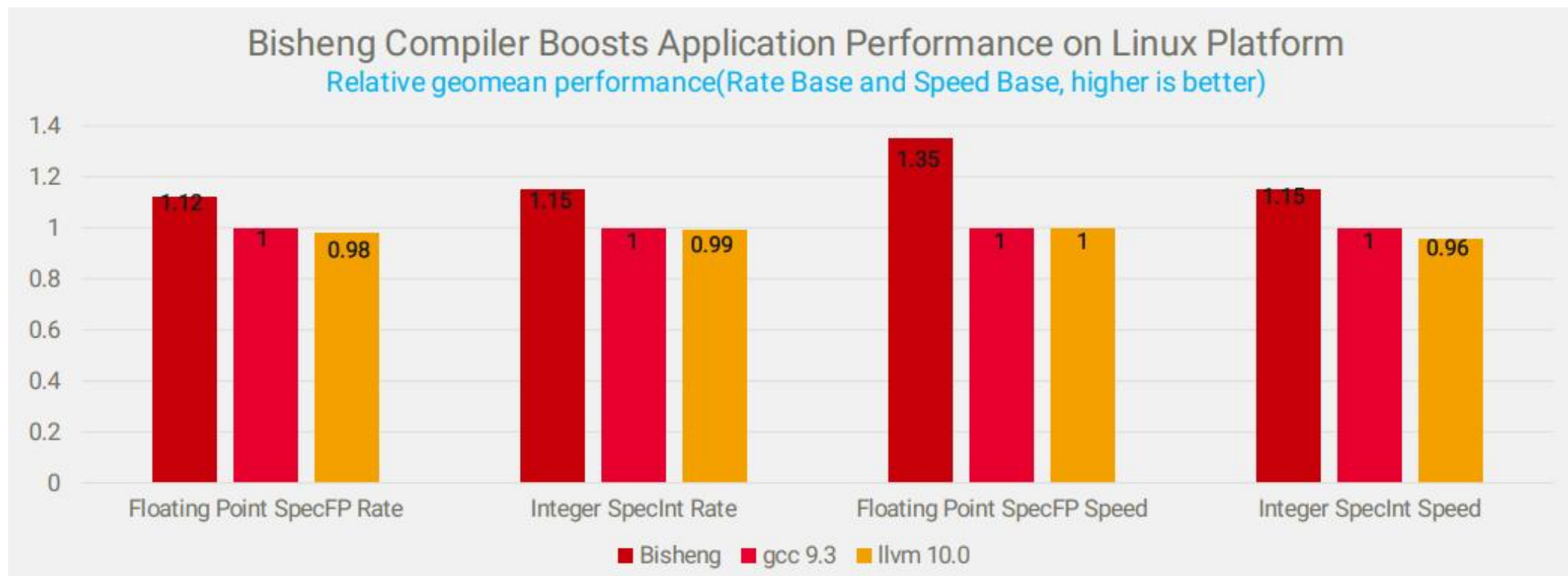
```
gloit@gloit-x1c > ~/2022_compiler_demo } master gcc -O2 add.c
gloit@gloit-x1c > ~/2022_compiler_demo } master ./a.out
Sum: 499999999500000000, Time Cost: 0.000002
```

性能提升数十万倍

国产开源编译器——毕昇编译器



- 毕昇编译器通过编译优化提升鲲鹏硬件平台上业务的性能体验，SPEC2017性能较业界编译器平均高15%以上。



SPEC作为业界芯片性能评分标准，SPEC的分数可以直观体现出硬件的性能，越高越好

□标准的指令式语言(Java, C, C++)

■ 状态

- 变量
- 结构
- 数组

■ 计算

- 表达式 (arithmetic, logical, etc.)
- 赋值语句
- 条件语句 (conditionals, loops)
- 函数

□ 状态

- 寄存器
- 内存单元

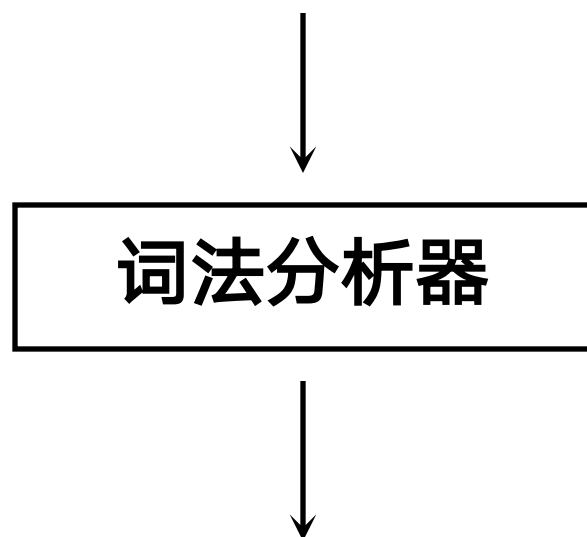
□ 机器码 – load/store architecture

- Load, store instructions
- 寄存器操作 – Arithmetic, logical operations
- 分支指令 – Branch instructions

□ 将程序字符流分解为记号 (Token) 序列

■ 形式: $\langle \text{token_name}, \text{attribute_value} \rangle$

position = initial + rate * 60 ← 字符流



符号表

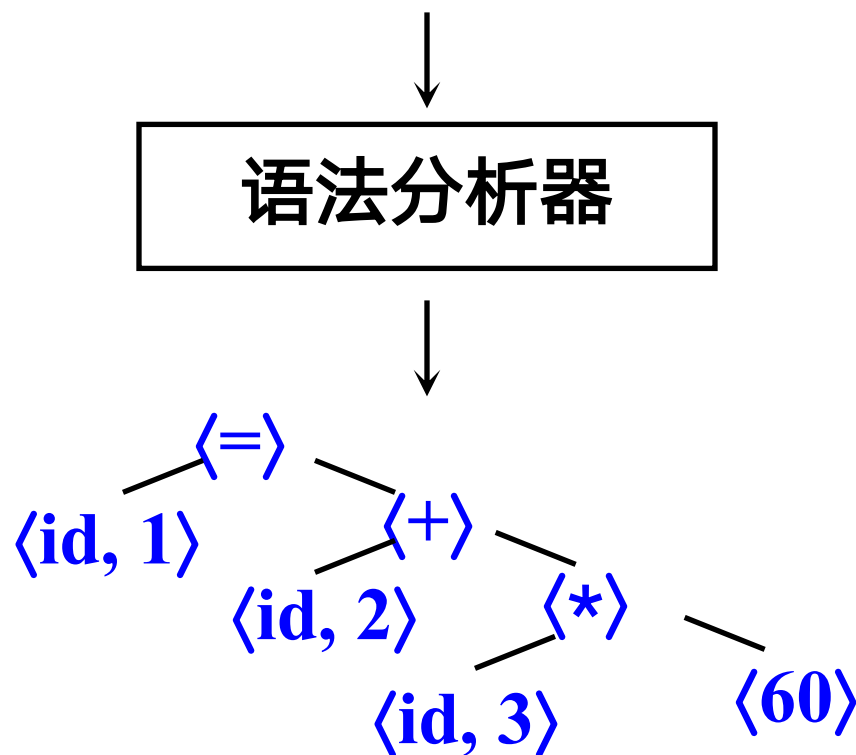
1	position	...
2	initial	...
3	rate	...

$\langle \text{id}, 1 \rangle \langle = \rangle \langle \text{id}, 2 \rangle \langle + \rangle \langle \text{id}, 3 \rangle \langle * \rangle \langle 60 \rangle$ ← 记号流

命令行输入: `clang -cc1 -dump-tokens xx.c`

□也称为解析（Parsing），在词法记号的基础上，创建语法结构

$\langle \text{id}, 1 \rangle \langle = \rangle \langle \text{id}, 2 \rangle \langle + \rangle \langle \text{id}, 3 \rangle \langle * \rangle \langle 60 \rangle \leftarrow$ 记号流



符号表

1	position	...
2	initial	...
3	rate	...

\leftarrow 语法树

命令行输入: `clang -fsyntax-only -Xclang -ast-dump xx.c`

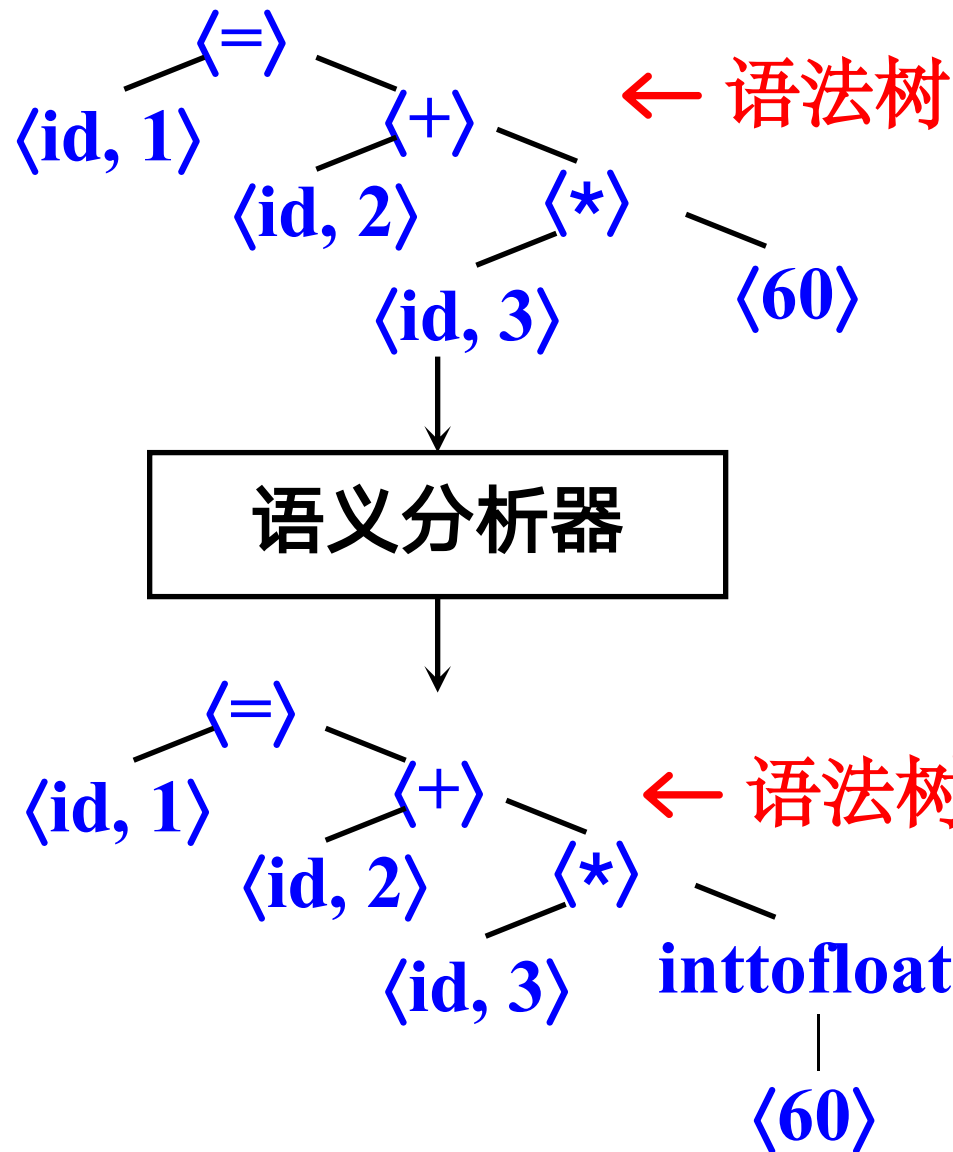
□ 编译器会检查程序中的不一致

■ 如：类型检查 (type checking)

符号表

1	position	...
2	initial	...
3	rate	...

注：类型转换在astdump时已经完成



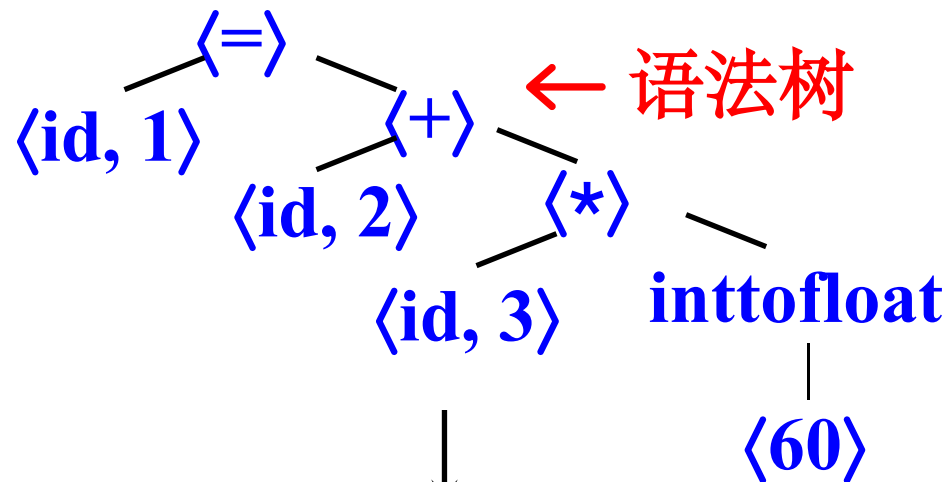
中间代码生成



□是源语言与目标语言之间的桥梁

符号表

1	position	...
2	initial	...
3	rate	...



中间代码生成器

$t1 = \text{inttofloat}(60)$

$t2 = \text{id3} * t1$

$t3 = \text{id2} + t2$ ← 中间代码

$\text{id1} = t3$

命令行输入: `clang -cc1 xx.c -emit-llvm -o xx.ll`

机器无关代码优化



- 机器无关的代码优化便于生成执行时间更快、更短或能耗更低的目标代码

符号表

1	position	...
2	initial	...
3	rate	...

```
t1 = inttofloat(60)
```

```
t2 = id3 * t1
```

```
t3 = id2 + t2
```

```
id1 = t3
```

← 中间代码



代码优化器



```
t1 = id3 * 60.0
```

```
id1 = id2 + t1
```

← 中间代码

命令行输入: `clang -S -emit-llvm -O3 test.c -o test-new.ll`

目标代码生成



- 如果目标语言是机器代码，必须为变量选择寄存器或内存位置

符号表

1	position	...
2	initial	...
3	rate	...

$t1 = id3 * 60.0$

$id1 = id2 + t1$ ← 中间代码



代码生成器



LDF R2, id3

MULF R2, R2, #60.0

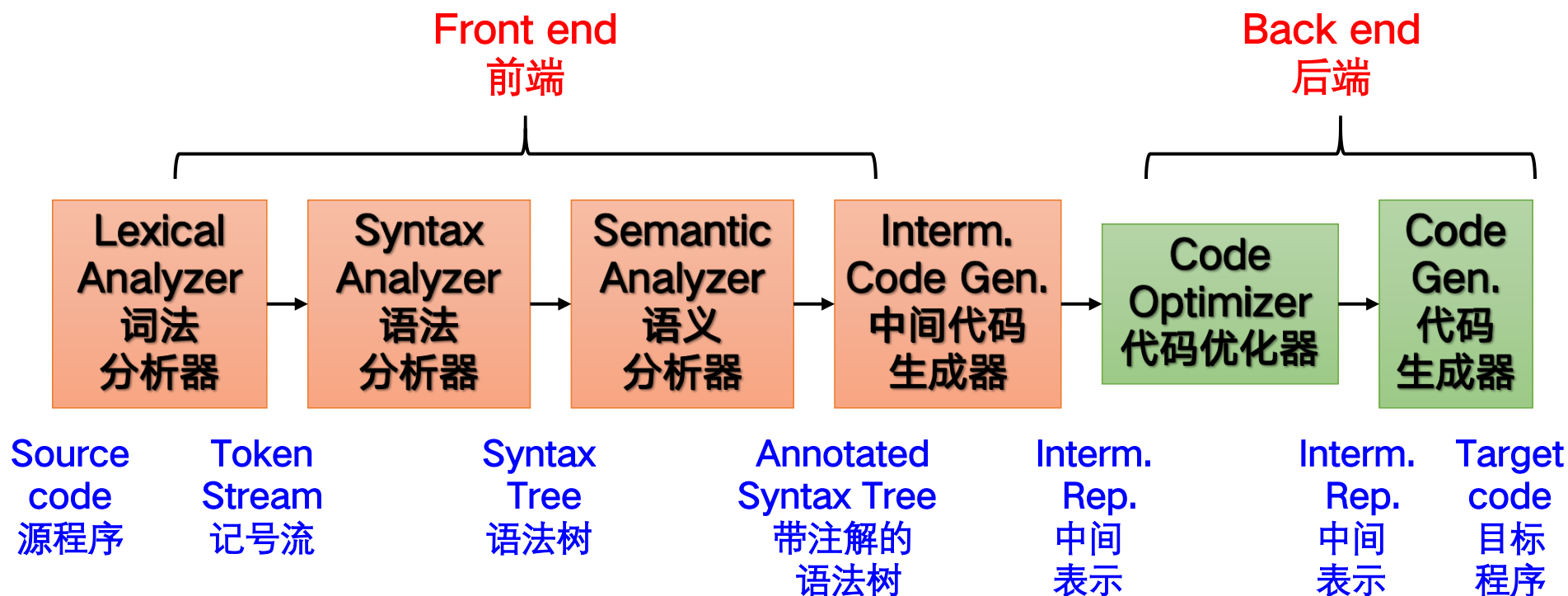
LDF R1, id2 ← 汇编代码

ADDF R1, R1, R2

STF id1, R1

命令行输入: llc-14 xxx.ll -o xxx.s

现代编译器的构造/阶段



Symbol Table 符号表

Error Handler 错误处理

1

编译器是什么？

2

为什么要学习编译课程？

3

编译教学的困境与科大方案

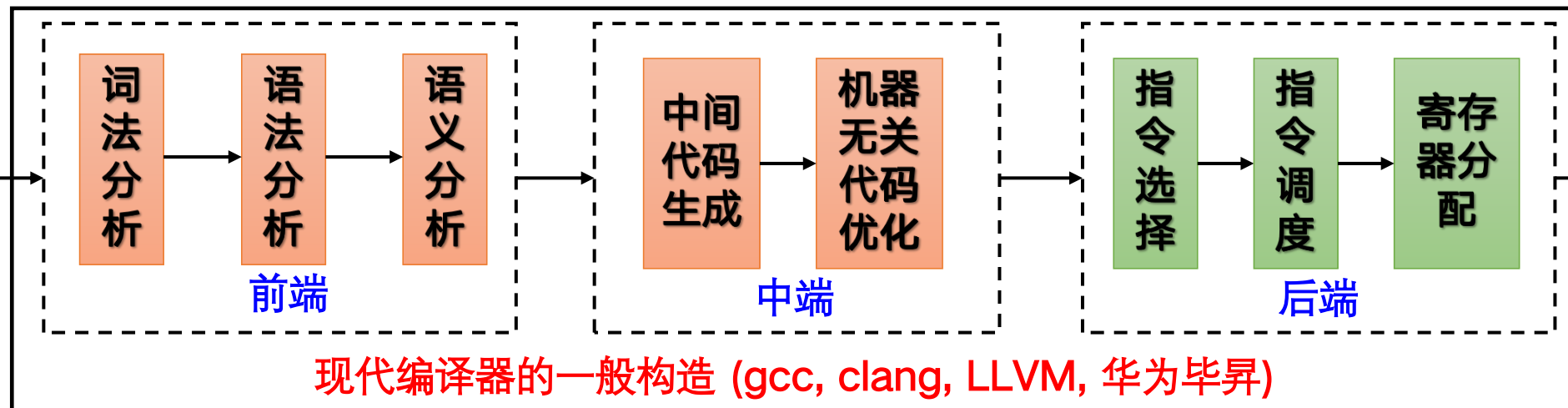
4

本学期课程设置的情况

编译课程简介



程序员编
写的
源程序



机器硬件上
运行的
目标代码



“编写**编译器的原理和技术**具有普遍的意义，以至于在每个计算机科学家的**研究生涯**中，该书中的原理和技术都会**反复用到**。”

——著名计算机专家 *Alfred V. Aho*

“在供应链可控性上，仍存在**编译工具依赖国外**的情况。”

“我们应重点**突破**操作系统内核、**编译器**等关键技术。”

——《中国信息技术产品安全可控年度发展报告》

编译技术历史悠久影响深远



□图灵奖自1966年颁发以来，共有75名获奖者，其中编译相关的科研人员有21位，占比28%，Alan J. Perlis因编译技术贡献成为第一位获得图灵奖的科学家。

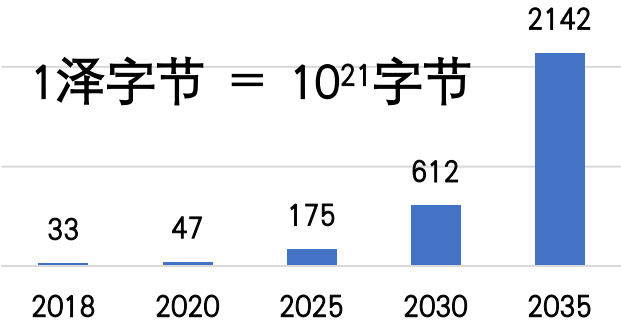
年份	科学家	贡献	年份	科学家	贡献
1966	Alan J. Perlis	高级程序设计技巧，编译器构造	1972	Edsger Dijkstra	程序设计语言的科学与艺术
1974	Donald E. Knuth	算法分析、程序设计语言的设计、程序设计	1976	Michael O. Rabin Dana S. Scott	非确定性自动机
1977	John Backus	高级编程系统，程序设计语言规范的形式化定义	1979	Kenneth E. Iverson	程设语言和数学符号，互动系统的设计，程设语言的理论与实践
1987	John Cocke	编译理论，大型系统的体系结构，及RISC计算机的开发	2005	Peter Naur	Algol 60语言
2006	Frances E. Allen	优化编译器	2020	Jeffrey David Ullman Alfred Vaino Aho	推进编程语言实现的基础算法和理论、教材撰写

编译技术历史悠久影响深远



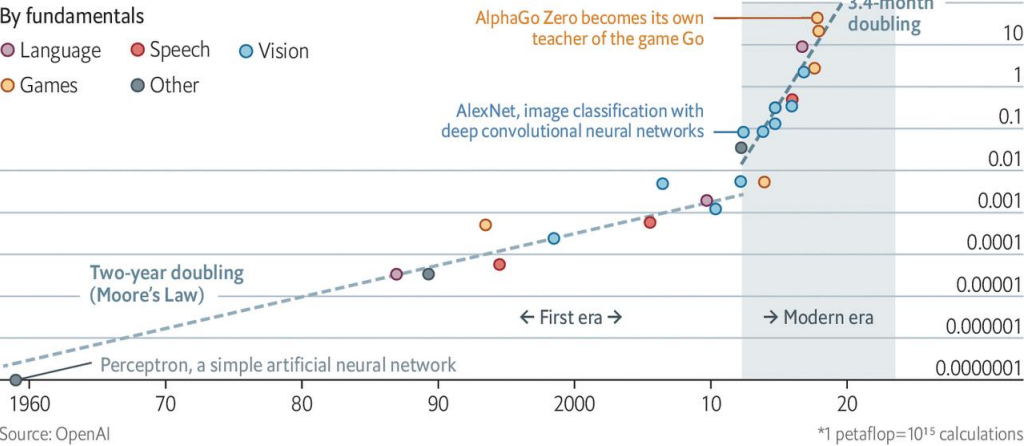
□图灵奖自1966年颁发以来，共有75名获奖者，其中编译相关的科研人员有21位，占比28%，Alan J. Perlis因编译技术贡献成为第一位获得图灵奖的科学家。

年份	科学家	贡献	年份	科学家	贡献
1980	C. Antony R. Hoare	程序设计语言的定义与设计	1983	Ken Thompson Dennis M. Ritchie	UNIX 操作系统和C语言
1984	Niklaus Wirth	程序设计语言设计、程序设计	2001	Ole-Johan Dahl Kristen Nygaard	面向对象编程
2003	Alan Kay	面向对象编程	2008	Barbara Liskov	编程语言和系统设计的实践与理论
2021	Jack J. Dongarra	通过对线性代数运算的高效数值算法、并行计算编程机制和性能评估工具的贡献，引领了高性能计算的世界。			

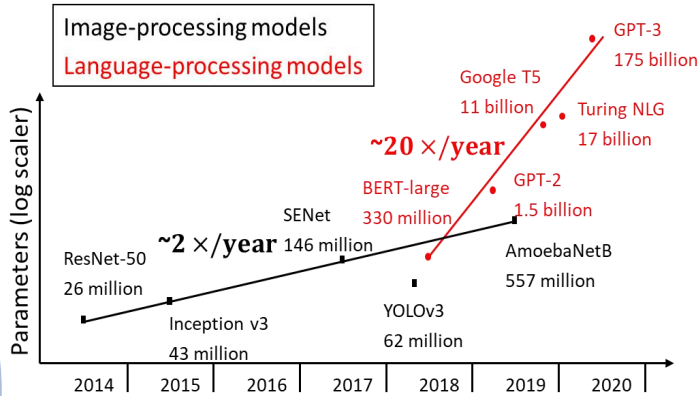


数字经济、数字孪生助推
全社会数据的爆炸式增长

Computing power used in training AI systems
Days spent calculating at one petaflop per second*, log scale

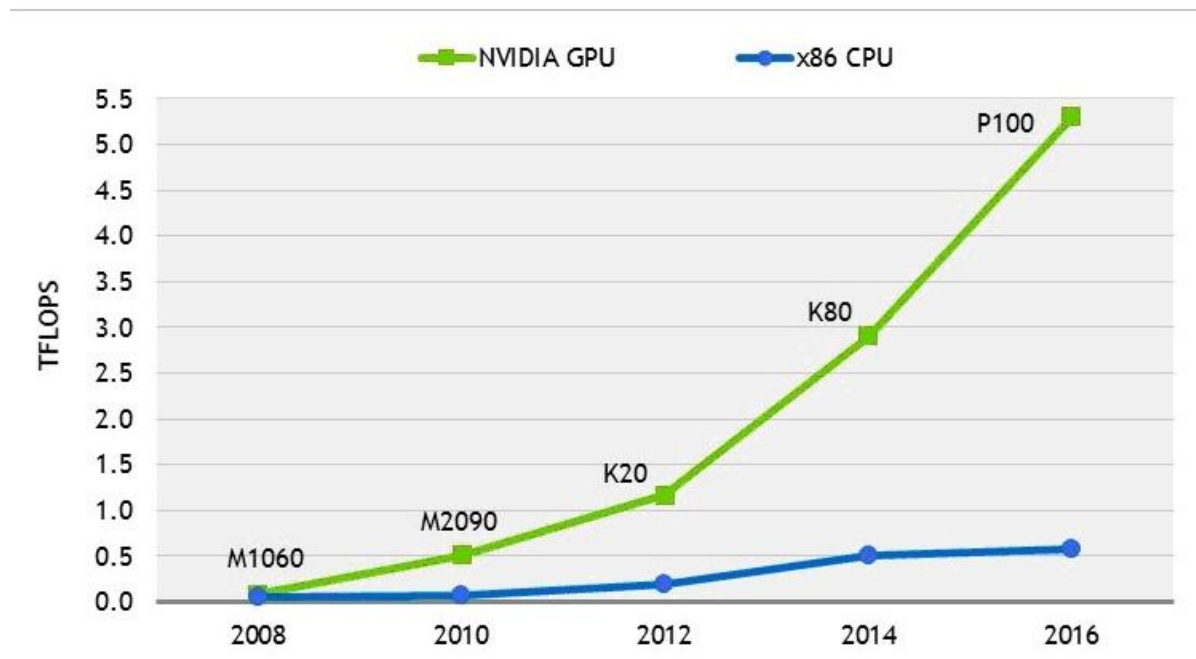
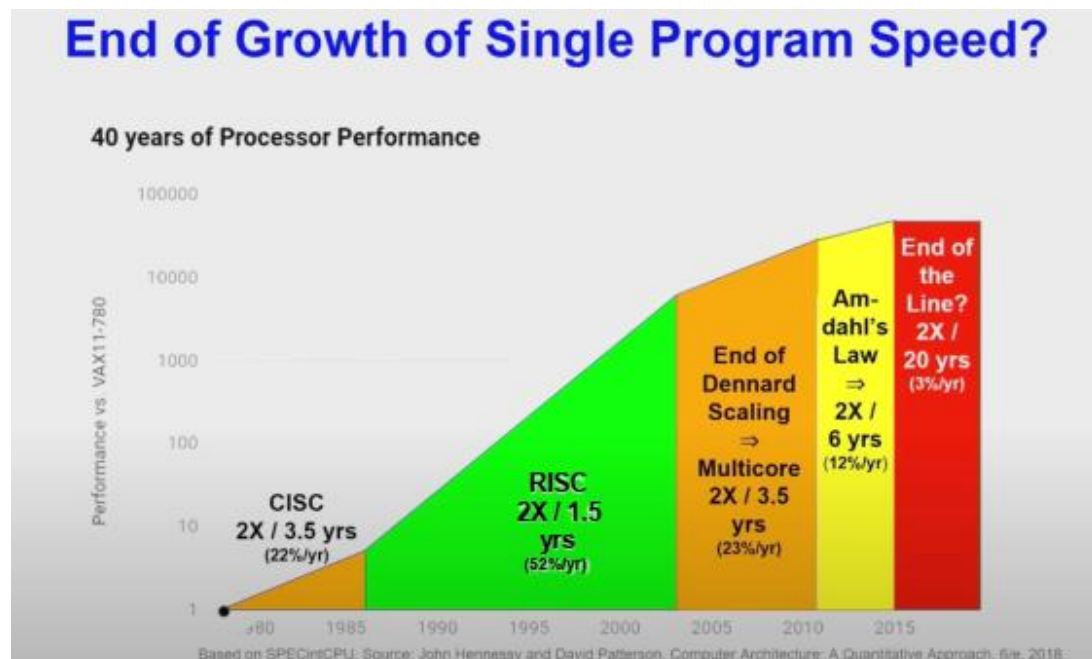


过去十年，AI训练算力需求增长了30万倍，3.4个月翻一倍



新版摩尔定律：
宇宙中的智能数量每18个月就会翻一倍

体系结构的黄金时代



Turing Lecture, Hennessy, Patterson; June 2018 / CACM Feb 2019

<https://www.rtinsights.com/gpus-the-key-to-cognitive-computing/>

❑ 领域专用芯片的使用助推了人工智能的发展

编译是工业界关注的焦点



❑ A New Golden Age for Computer Architecture

■ By John L. Hennessy, David A. Patterson; Communications of the ACM

❑ 新硬件推陈出新的时代

■ GPU、DPU、TPU、NPU、xPU

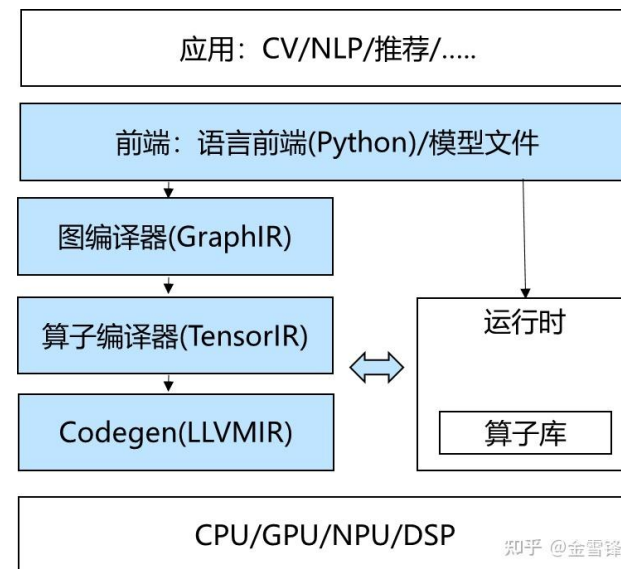
■ 量子计算机

❑ 新应用不断涌现的时代

■ AI计算、科学计算 (AlphaFold)、量子计算

❑ 编译技术的需求日益旺盛

■ 华为方舟、毕昇编译器、龙芯二进制翻译、AI编译器 (XLA、TVM)



体系结构黄金时代的机会



□ 软件为中心

- 现代编程语言很多是script语言-> 需要解析、动态类型、代码复用
- 编程高效但是执行不见得高效: Python vs. C

□ 硬件为中心

- 唯一可行的路径是领域专用架构
- 一颗芯片只做一部分事情, 但是性能很好

□ 软硬协同的设计思路

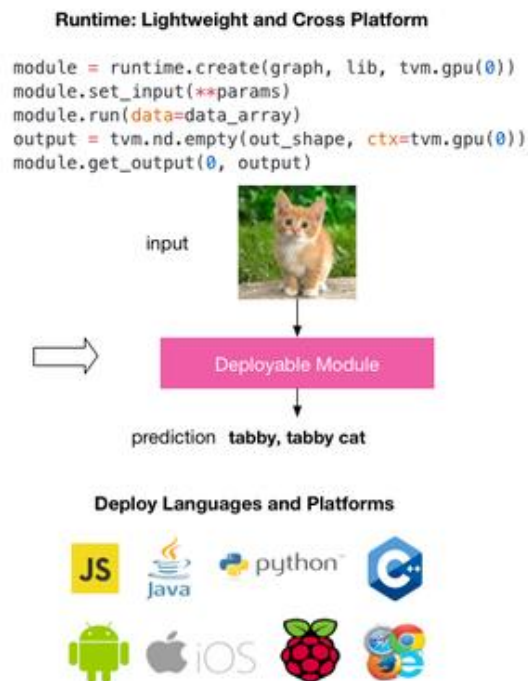
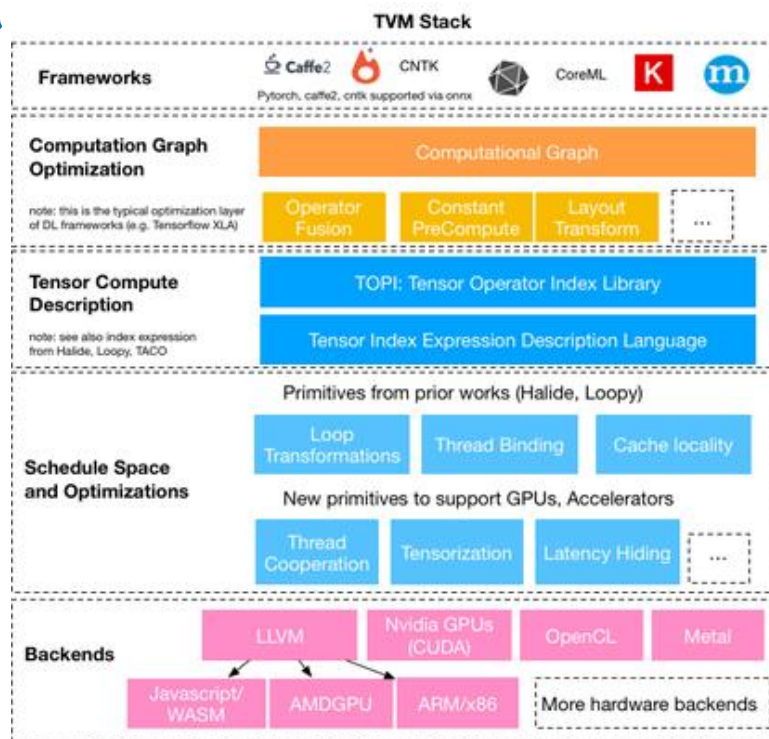
- 专用领域语言+专用领域架构: CUDA + GPU
- 可以最大化挖掘硬件的并行度

新一代编译技术呼之欲出



□ 新一代编译器需要具备以下基础能力

- 为异构芯片提供统一的硬件抽象
- 支持为异构芯片生成对应的目标代码
- 能理解专用领域的语言和编程范式
- 编译框架自身的质量、可靠性和性能



产业界编译人才紧缺



编译器 [杭州·滨江区·长河] 25-50K 经验不限 本科 刘女士 编译器与芯片... Go 后端开发 加班补助, 补充医疗保险, 定期体检, 五险一金, 年终奖, ...	 华为 计算机软件 不需要融资 10000人以上
编译器开发 [西安·雁塔区·高新软件园] 20-40K·16薪 经验不限 硕士 陈先生 研发工程师 编译器 Loongarch 龙芯 定期体检, 住房补贴, 生日福利, 员工旅游, 包吃, 补充医...	 龙芯中科 电子/半导体/集成电路 已上市 500-999人
编译器工程师 [上海·浦东新区·金桥] 18-35K 1-3年 本科 王先生 软件研发工程师 数据结构 架构师 ARM开发 住房补贴, 定期体检, 离职补偿, 零食下午茶, 补充医疗保...	 华为技术有限公司 计算机软件 不需要融资 10000人以上
编译器工程师 [上海·徐汇区·徐家汇] 20-30K·15薪 经验不限 本科 韩女士 技术经理 算法基础 Linux 编译工具链 汇编 反汇编 年终奖, 免费班车, 员工旅游, 定期体检, 五险一金, 员工...	 麒麟软件 计算机软件 未融资 1000-9999人

职位信息

职位描述:

- 1、负责基于编译技术的字节跳动核心业务的性能分析及优化;
- 2、负责各种编程语言(C++/Java/Go/Rust/WebAssembly)及周边基础库在业务中优化、技术推广与生态建设;
- 3、负责AutoFDO相关优化落地的研发维护;
- 4、负责前沿编译优化技术在字节跳动核心业务的落地;
- 5、负责编译技术针对X86_64, ARM64, RISC-V等体系结构的功能开发与性能优化;
- 6、负责异构编译技术针对AI领域与异构硬件的功能开发与性能优化;
- 7、参与研发业界领先的性能分析及优化平台。

职位要求:

- 1、熟悉编译原理以及相关编译优化技术;
- 2、熟悉LLVM/GCC/GraalVM/OpenJDK/Go中某个编译器框架或者Runtime;
- 3、熟悉C++/Java/Go/Rust/WebAssembly中某个语言的设计与实现原理;
- 4、熟悉Intel/AMD x86_64或ARM64、RISC-V体系结构, 精通x86_64、ARM64、RISC-V微架构;
- 5、精通C/C++, 熟悉C++底层实现原理;
- 6、有以下经验者优先:

- a. 熟悉LLVM/GCC等C/C++编译器, 并参与相关社区者优先;
- b. 有HPC/AI编译器以及高性能库开发经验者优先。

职能类别: 系统架构设计师

1

编译器是什么？

2

为什么要学习编译课程？

3

编译教学的困境与科大方案

4

本学期课程设置的情况

编译教学及人才培养目标



计算机专业核心课 编译原理和技术

编译器

- ★ 一般构造原理
- ★ 基本设计方法
- ★ 主要实现技术

有助于培养学生的：

- 计算机系统能力
- 综合、创新能力
- 软件工程能力

目标1：从事编译相关产业的领军人才

目标2：从事编译相关产业的技术骨干

目标3：使用编译技术解决其他关键问题的人才

智能时代高性能编译人才培养体系建设



国家需求

- 高性能芯片禁运，国产芯片突围
- 编译尖端人才储备与产能不足
- “打好基础软件国产化攻坚战”



中华人民共和国中央人民政府

www.gov.cn

习近平主持中共中央政治局第三次集体学习并发表重要讲话

2023-02-22 12:30 来源：新华社

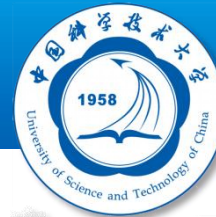
建设思路

围绕**自主可控**芯片技术，突出**工程实践训练**
培养具备**创新精神**和**系统能力**的**复合型高性能计算人才**

教改项目

教育部产学研合作协同育人	基于国产自主指令集的编译实验慕课建设	负责人
教育部供需对接就业育人	大模型分布式高效并行训练方法研究	负责人
安徽省省级教学研究	以系统能力培养为导向的编译实验教学改革	负责人

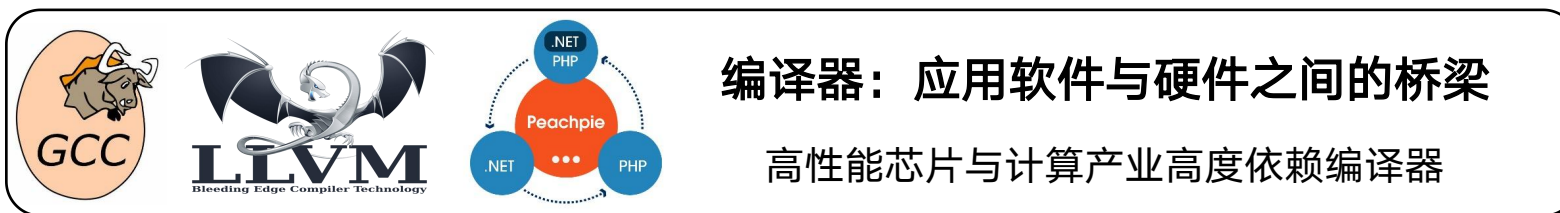
国内外实践体系调研



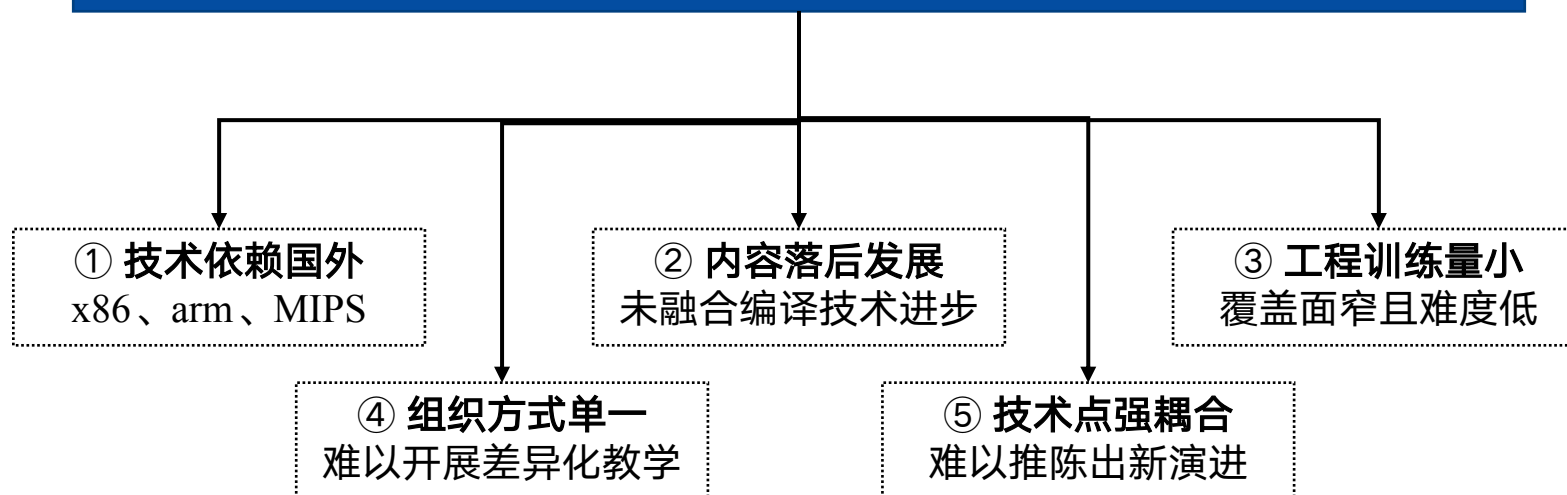
		前端语言类型	词法器	语法器	中间代码生成	后端代码生成	中间代码优化	寄存器分配
国外高校	ETH		√	√	LLVM	x86	√	√
	剑桥	基于Ocaml	√	√	Ocaml IL	ARM、RISC-V	√	√
国内高校	清华大学(交叉院)	C/C++/Python	√	√	√	RISC-V	×	√
	哈工大		√	√	√	×	×	×
	东北大学	PASCAL	√	√	四地址码	x86	×	×
	东北师大	×	×	×	×	×	×	×
	中科大	C/C++	√	√	LLVM	LoongArch	√	√

□部分高校未开展编译实践；部分学校未贯通编译实践的全部环节；LLVM重要但未被广泛采用；x86和RISC-V仍是主流；国产自主体系未体现

编译重要性与人才培养挑战



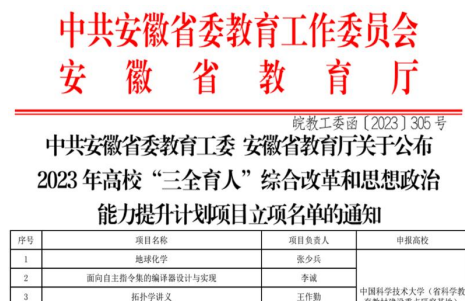
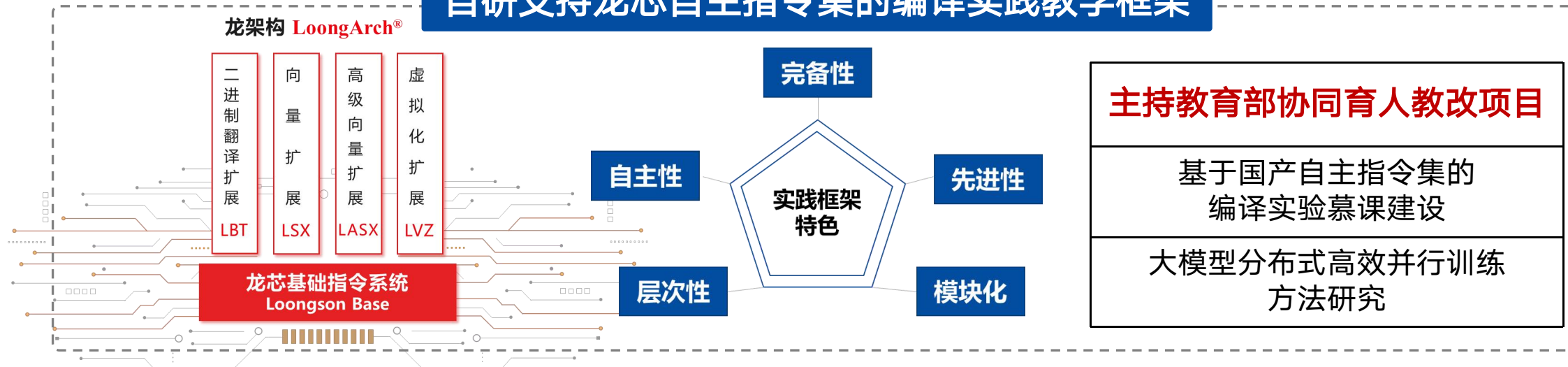
编译人才培养产能不足，根源在于实验体系的缺失错位



教学实践：建设基于自主芯片技术的编译课程



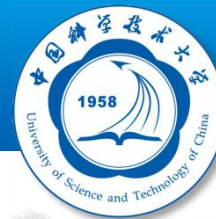
自研支持龙芯自主指令集的编译实践教学框架



实验教材出版合同已签订，入选**安徽省精品教材**计划

第二批国家级线下一流本科课程
(2076门中仅两门编译课程)

教学实践：全国范围内的编译课程推广



暑期系统能力师资培训导教班



西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

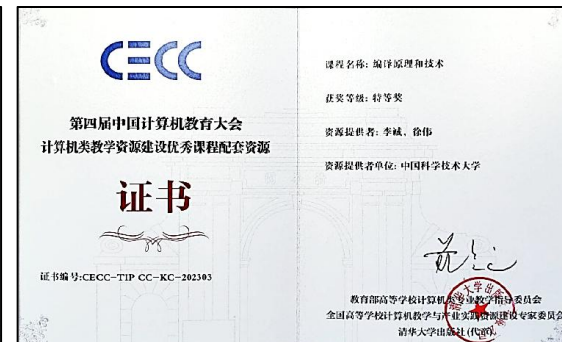
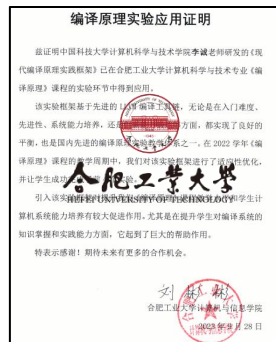


四川大学
SICHUAN UNIVERSITY



兰州大学
LANZHOU UNIVERSITY

在教指委系统能力专家组指导下，开设两期导教班
学员来自西工大、川大、合工大等30所高校



教学资源已被**湖南大学、合工大等24所高校**
师生使用，覆盖校外学生**3500+**

第四届中国计算机教育大会优秀教学资源
2项特等奖（包揽编译方向全部奖项）

Q



5485 0 2024-02-06 11:08:52 未经授权，禁止转载

编译课程资源: <https://ustc-compiler-principles.github.io/textbook/>

计算机技术

38.5万 2679

36

1

编译器是什么？

2

为什么要学习编译课程？

3

编译教学的困境与科大方案

4

本学期课程设置的情况

课程设置



□每周四(6,7,8) , 高新区GT-B109教室

□课程信息化平台 (1个网站1个群) :

■课程主页:

➢ <https://USTC-Compiler-2025.github.io/CompEng-homepage>

➢ 发布讲义

■qq群 (440229957)

➢ 发布通知、非灌水群



□教材和参考书

- 陈意云、张昱, 编译原理 (第3版), 高等教育出版社, 2014
- A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman 著, 赵建华等译, 编译原理, 机械工业出版社, 2017

□实验视频

- B站视频: <https://www.bilibili.com/video/BV1D7421N7c4>

□其他资料

- Stanford课程主页
<http://web.stanford.edu/class/cs143/>
- MIT课程主页:
<http://6.035.scripts.mit.edu/fa18/>

考核要求



□考核内容包括平时考核、工程实践

□成绩组成：

■平时考核（50-65%）

- 按时上课（特殊情况不能来需要书面请假）
- 按时完成作业

■工程实践（35-50%）

- 选择优化pass，根据算法分析LLVM的具体实现过程等实验

■该比例会动态调整，最终解释权归课程组老师所有！

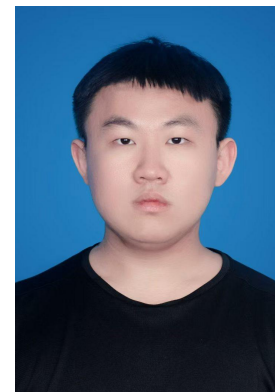
助教团队



徐伟 博士
高级实验师



刘睿博
大四保研



李宇哲
大四保研



一起努力 打造国产基础软硬件体系！

李 诚、徐伟

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

先进技术研究院、计算机科学与技术学院

2025年02月27日