

# Visão Geral da Compilação

## Introdução

**Sandro Rigo**

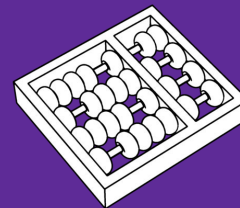
sandro@ic.unicamp.br

*Universidade Estadual de Campinas (Unicamp)*

*Instituto de Computação (IC)*

*Laboratório de Sistemas de Computação (LSC)*

**MC921** • Projeto e Construção de Compiladores • 2024 S1



UNICAMP

# Plano

- Organização da Disciplina
- O que faz um compilador?
- Historia dos compiladores
- O que é um compilador?
- Estrutura dos compiladores

# Organização da Disciplina



- Carga horaria
  - 90H
- Creditos
  - 6
- Aulas
  - Terça 08-10H (PB14)
  - Quinta 08-10H (PB14)
- Laboratorios
  - A: Ter 10-12H (CC00)
  - B: Qui 10-12H (CC00)
- Monitores
  - César Carneiro
  - Vitoria Dias
- Slides gentilmente cedidos pelo Prof Hervé

1. Visão geral da compilação
2. Análise léxica: Lexer
3. Análise sintática: Parser
4. Análise semântica
5. Otimização de código
6. Geração de código
7. Compiladores no mundo real


- Aulas
  - Introdução dos conceitos
  - Exercícios
- Trabalhos teóricos
  - Avaliações via Google Classroom
  - Seminários
- Projetos praticos

$$MP = 0.1 P1 + 0.2 P2 + 0.2 P3 + 0.2 P4 + 0.3 P5$$

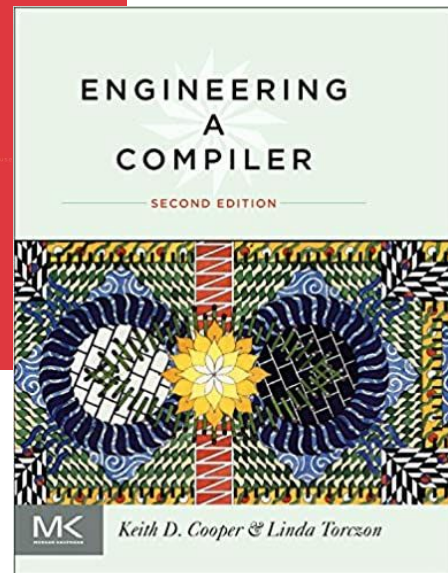
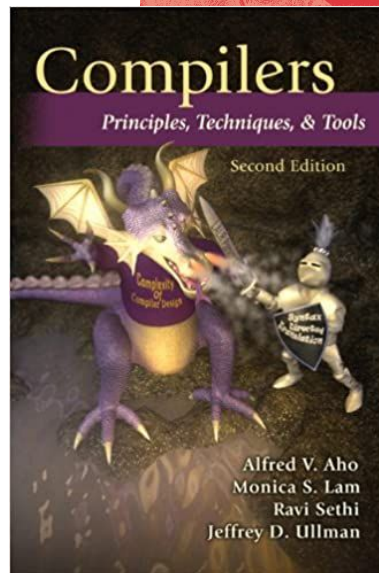
MT = Média aritmética dos trabalhos teóricos e seminário

$$M = 0.7 MP + 0.3 MT$$

se MP e MT acima de 5, caso contrário,  $M = \min(MP, MT)$

- Construção de um compilador para a linguagem uC
  - Trabalho em grupo (max 2 pessoas)
  - Desenvolvido em Python
  - Dividido em 5 projetos
  - Cada novo projeto precisa dos projetos anteriores
  - Dificuldade e peso crescente
- Usarão Github Classroom
  - Repositório base fornecido
  - Notebook com as instruções
- Avaliação
  - Cada projeto tem um conjunto de testes fornecido (pytest)
  - Todos os testes são abertos
  - Detecção automática de fraude 

- Andrew Appel.  
**Modern Compiler Implementation in Java.**
- Aho, Sethi and Ullman.  
**Compilers: Principles, techniques and tools.**
- Keith Cooper and Linda Torczon.  
**Engineering a Compiler.**





# O que faz um Compilador?

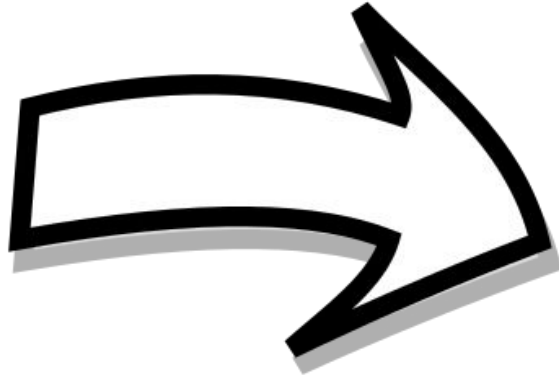


# O que faz um compilador?

10



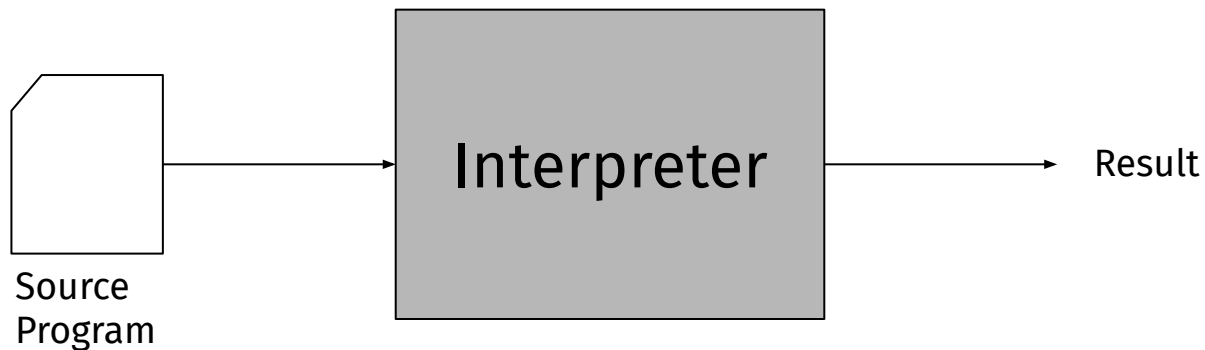
Linguagens de  
Programação



Hardware



- O compilador traduz o programa de uma linguagem para outra
  - Em geral, a fonte é linguagem de programação e o alvo é o código de máquina
  - Algumas exceções (Source-to-source compiler, dynamic binary translation, etc)
- O compilador verifica a conformidade da entrada

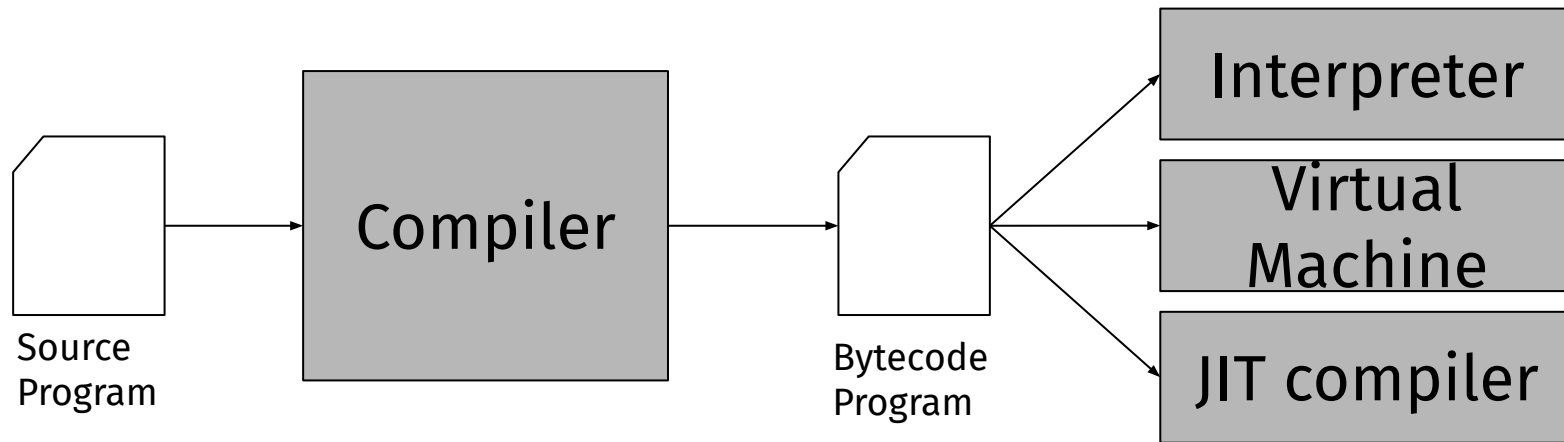


- O interpretador produz um resultado
  - tem várias similaridades com o compilador

- Quais linguagens são interpretadas?
  - PHP, Ruby, Python, JavaScript...
- Quais linguagens são compiladas?
  - C/C++, Rust, Go, C#, Java...
- Vantagem da interpretação:
  - Executa direto
  - Portabilidade
- Vantagem da compilação:
  - Velocidade de execução

Não é  
tão simples!

Muitas linguagens  
são **compiladas** e  
**interpretadas** hoje



- O compilador traduz o programa em bytecode
- O bytecode pode ser
  - Interpretado pelo interpretador (Python, etc)
  - Executado/Interpretado pela máquina virtual (Java, etc)
  - Compilado pelo compilador JIT em código de máquina (Java, C#, etc)

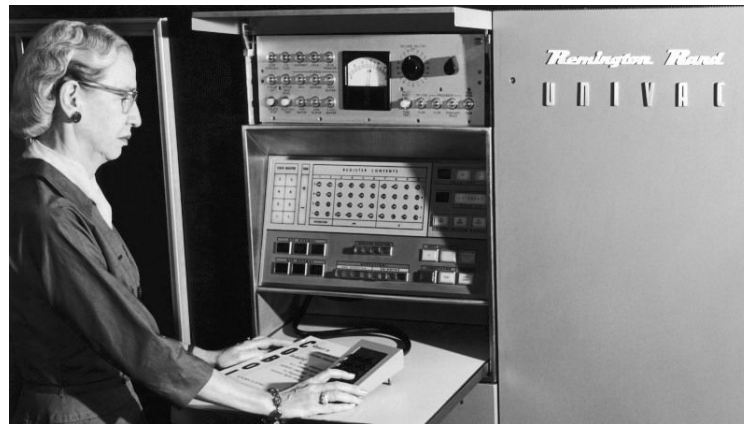
# História dos Compiladores



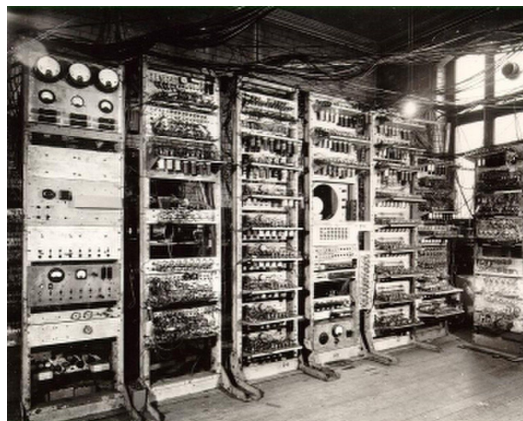
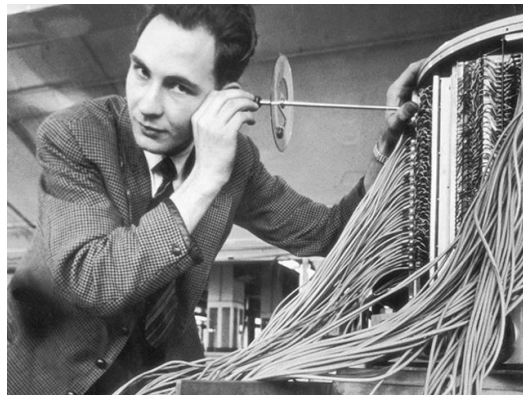
- Por muitos anos, o software dos **primeiros computadores** era escrito principalmente em **linguagem de montagem**
- As **linguagens de alto nível** de programação não foram inventadas até que os benefícios de ser capaz de **reutilizar software** em diferentes tipos de processador passassem a ser significativamente maiores do que o custo de **desenvolver um compilador**
- A capacidade de memória muito limitada dos primeiros computadores também criava muitos problemas técnicos na implementação de um compilador



- Escrito por Grace Hopper
  - Almirante da Marinha dos EUA
- Sistema A-0
  - Arithmetic Language version 0
  - Para o UNIVAC I
- Programa especificado como sequência de sub-rotinas e argumentos
  - Convertido em código de máquina

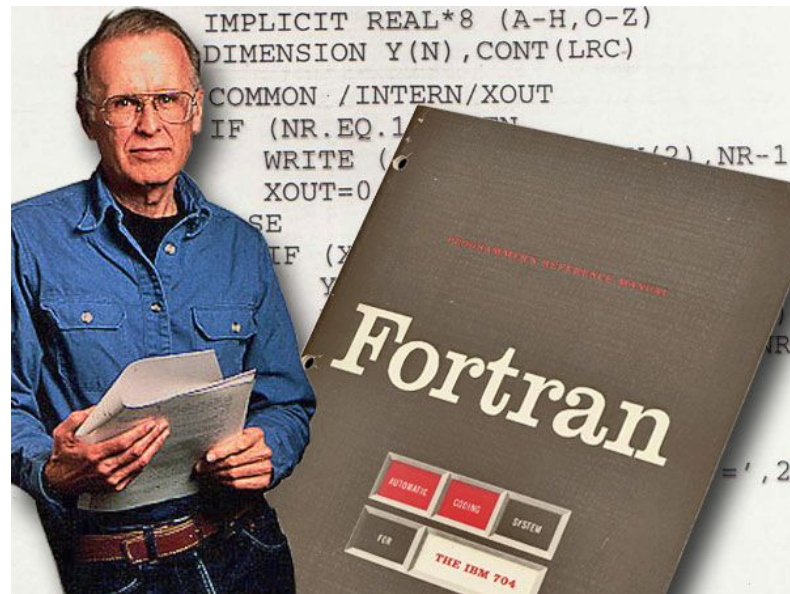


- Escrito por Alick Glennie
  - um cientista britânico
  - Na Universidade de Manchester
- Autocode
  - Facilitar a programação do Manchester Mark-1
- Glennie trabalhou com Alan Turing em vários projetos

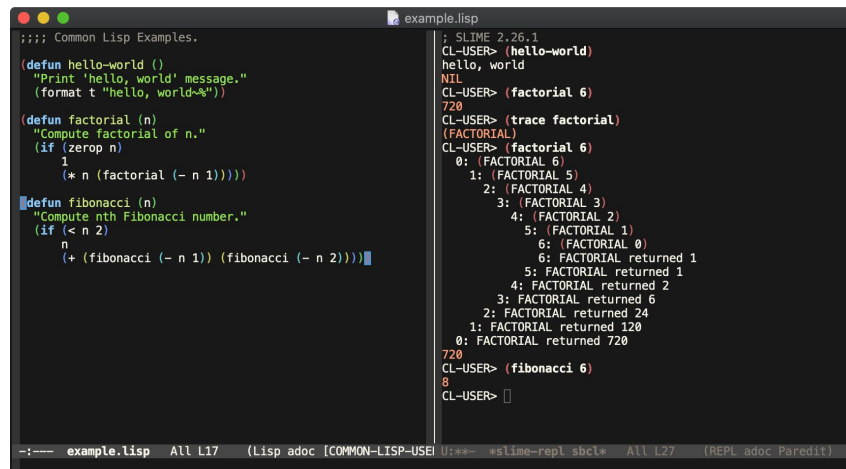


- Linguagens experimentais (até 1957)
  - Short Code no UNIVAC, Speedcoding no IBM 70, Whirlwind, BACAIC, PRINT, etc
- Fortran
  - Criado em 1954 e muito usado em computação científica
  - Última versão é Fortran 2018
- COBOL
  - Criado em 1959 e sucesso comercial
  - Successor do B-0 (1957) e do Flow-Matic (1958) da Grace Hopper
- Outras linguagens
  - Lisp (1958)
  - ALGOL (1958) -> primeira linguagem estruturada

- Em 1957, primeiro compilador completo para o FORTRAN
  - A equipe da IBM liderada por John Backus
  - Levou 18 person/years de trabalho
  - Primeiro compilador otimizador
  - Base de muitas técnicas clássicas para análise e otimização de código



- Os primeiros compiladores foram escritos em linguagens de montagem
- Um compilador pode ser escrito no próprio código-fonte que compila
  - Em 1958, o Laboratório de Eletrônica da Marinha dos EUA desenvolveu um compilador na linguagem ALGOL
  - Em 1962, Tim Hart e Levin Mike (MIT), criaram um para o Lisp



```
;;; Common Lisp Examples.

(defun hello-world ()
  "Print 'hello, world' message."
  (format t "hello, world~%"))

(defun factorial (n)
  "Compute factorial of n."
  (if (zerop n)
      1
      (* n (factorial (- n 1)))))

(defun fibonacci (n)
  "Compute nth Fibonacci number."
  (if (< n 2)
      n
      (+ (fibonacci (- n 1)) (fibonacci (- n 2)))))

; SLIME 2.26.1
CL-USER> (hello-world)
hello, world
NIL
CL-USER> (factorial 6)
720
CL-USER> (trace factorial)
(FACTORIAL)
CL-USER> (factorial 6)
0: (FACTORIAL 6)
1: (FACTORIAL 5)
2: (FACTORIAL 4)
3: (FACTORIAL 3)
4: (FACTORIAL 2)
5: (FACTORIAL 1)
6: (FACTORIAL 0)
6: FACTORIAL returned 1
5: FACTORIAL returned 1
4: FACTORIAL returned 2
3: FACTORIAL returned 6
2: FACTORIAL returned 24
1: FACTORIAL returned 120
0: FACTORIAL returned 720
720
CL-USER> (fibonacci 6)
8
CL-USER>
```

**Quais compiladores  
são usados hoje?**

# O que é um Compilador?



# “Um compilador

é um programa de computador (ou um grupo de programas) que, a partir de um código fonte escrito em uma linguagem de programação, cria um programa semanticamente equivalente, porém escrito em outra linguagem.”

Fonte: Dragon Book



1. Preservar o significado do programa que está sendo compilado
  - Gera código corretamente
2. Melhorar o programa de entrada de alguma forma discernível
  - Conceito de “optimizing compiler”
  - Por exemplo: minimizar o tempo de execução, o espaço ocupado na memória, o tamanho do armazenamento e o consumo de energia

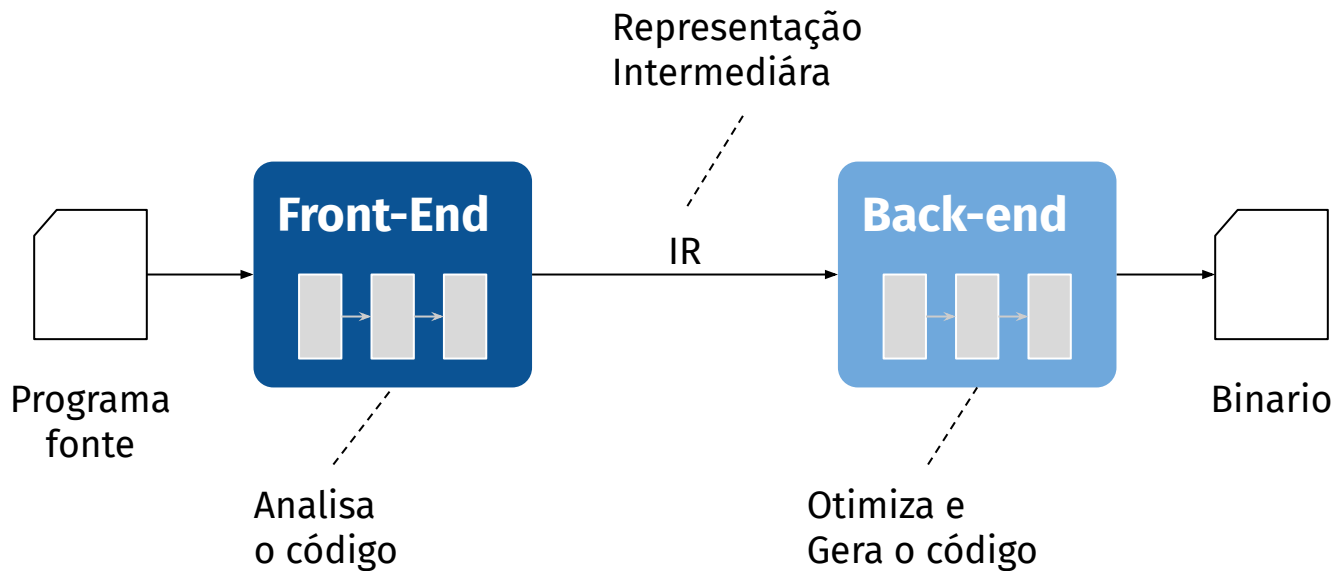
- Velocidade do código
  - Desempenho em tempo de execução do código compilado
- Feedback
  - Relatório quando encontrar um programa incorreto
  - User-friendliness se mede pela qualidade das mensagens de erros
- Depuração
  - Capacidade de usar um depurador *source-level*
- Eficiência em tempo de compilação
  - A compilação precisa ser rápida
- Espaço em memória
  - Tamanho do binário

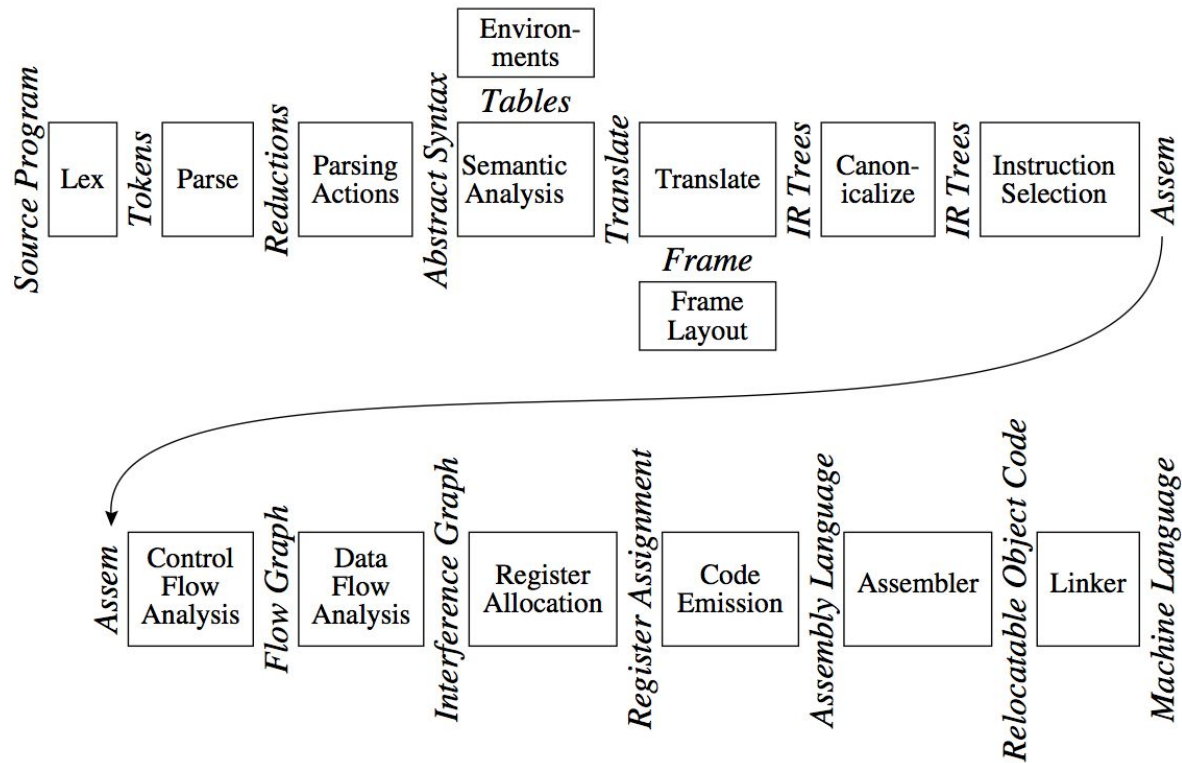
- São largamente usadas em muitas áreas
  - Linguagens de formatação de textos (latex, etc)
  - Compiladores *hardware* que transformam uma linguagem de especificação de circuitos VLSI em um projeto de circuitos
  - Editores de texto orientados a sintaxe
  - Analisadores estáticos de programa que podem descobrir variáveis não inicializadas, código que nunca será executado, etc
  - Checadores de entrada de dados em qualquer sistema
  - Interpretadores de comandos de um sistema operacional

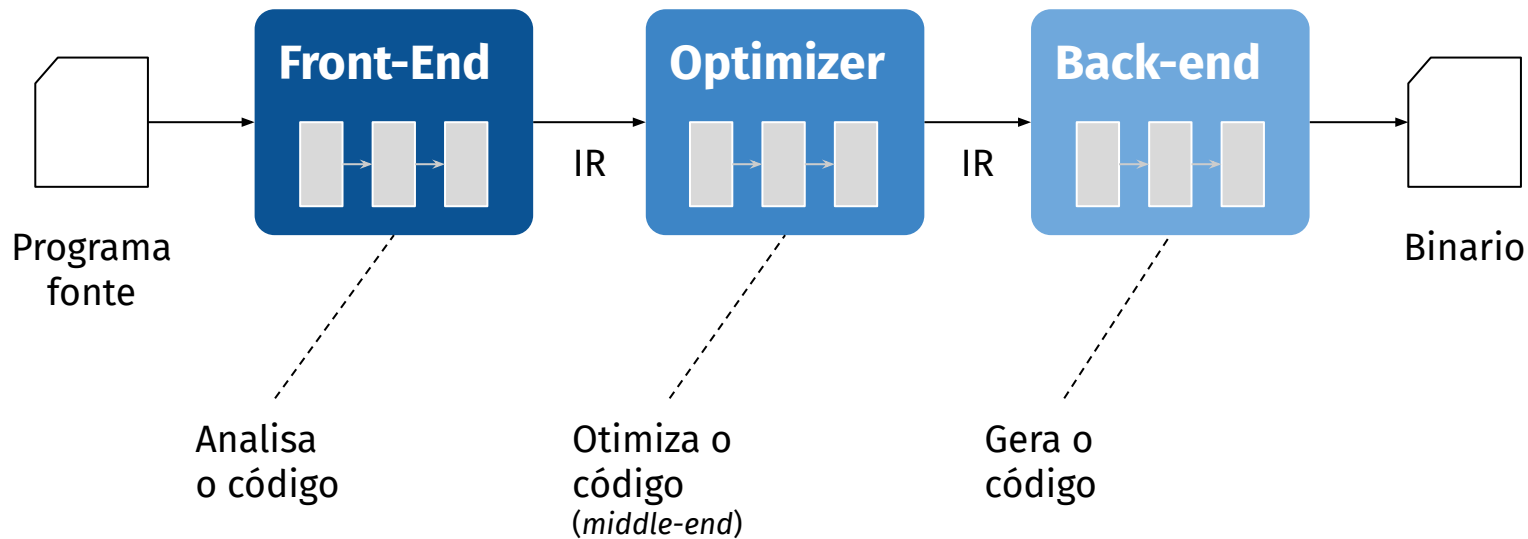
- Compiladores são software complexos
  - É um exercício interessante de engenharia de software
- Compiladores tem um papel fundamental em Ciência da Computação
  - A maior parte dos software são compilados
  - Entender compiladores ajuda a programar melhor
- O mercado precisa de experts em compiladores
  - Emprego em grandes companhias de software:  
Google, Apple, Facebook, Microsoft, Nvidia, Intel, AMD, ARM, etc
- Há ainda muitos problemas a serem resolvidos
  - Compilar modelos de Machine Learning
  - Compilar para aceleradores (GPU, etc)
  - Reduzir o consumo de energia

# Estrutura do Compilador

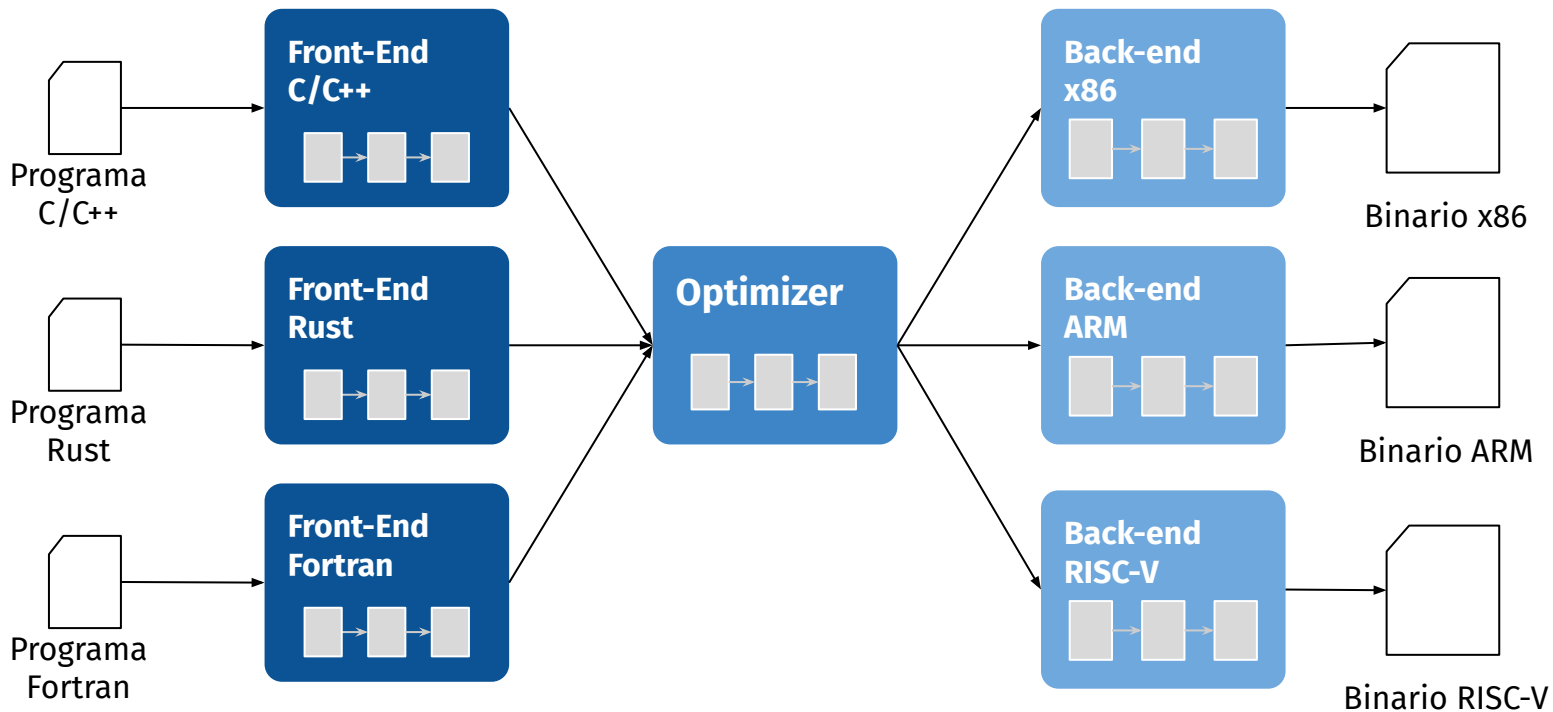












# Resumo

- Organização da Disciplina
- O que faz um compilador?
- Historia dos compiladores
- O que é um compilador?
- Estrutura dos compiladores

# Leitura Recomendada

---

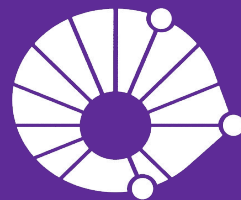
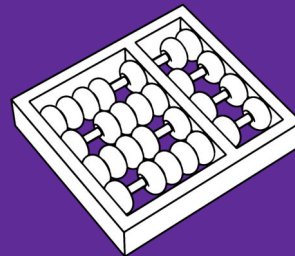
- Capítulo 1 do livro do Cooper.
- Capítulo 1 do livro do Appel.

# Proxima Aula

---

- Front-end do Compilador
  - Análise Léxica (Lexer/Scanner)
  - Como transformar caracteres em palavras?
  - Análise Sintática (Parser)
  - Como transformar palavras em frases?

**Obrigado!**  
**Merci!**



**UNICAMP**

# Pallette



**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**BUBBLE**

Lorem ipsum dolor  
sit amet, consectetur  
adipiscing elit.

**DRACULA**

Table Title	
Column 1	Column 2
One	Two
Three	Four

Table Title	
Column 1	Column 2
One	Two
Three	Four

Table Title	
Column 1	Column 2
One	Two
Three	Four

Table Title	
Column 1	Column 2
One	Two
Three	Four



