Compiladores

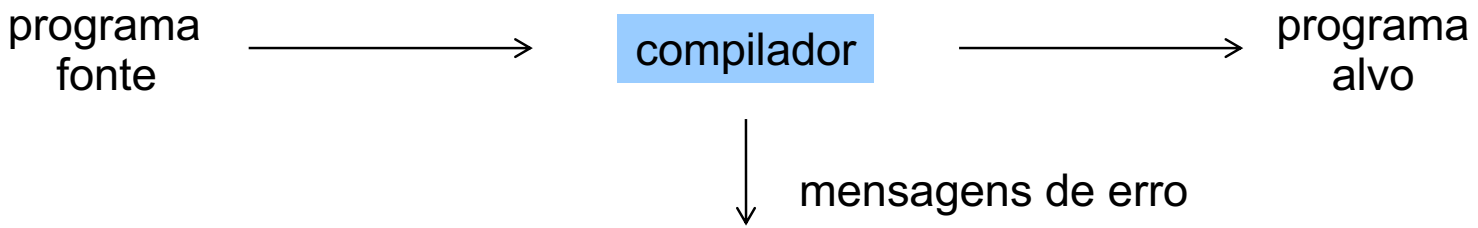
Licenciatura em Engenharia Informática

Universidade de Coimbra

Compiladores

O que é um compilador?

- Programa que lê outro programa escrito numa linguagem – a linguagem *fonte* – e o traduz num programa equivalente noutra linguagem – a linguagem *alvo*.
- Durante a tradução, o compilador reporta a eventual existência de erros no programa fonte



Compiladores

- Linguagem fonte
  - C/C++
  - Pascal
  - Fortran
  - Java
  - Go
  - ...
- Linguagem alvo
  - Assembly (de diversos sabores)
  - Bytecode

Compiladores

Como funciona um compilador?

- Análise
  - Da linguagem fonte à representação intermédia
  - Lexical, sintática, semântica
- Síntese
  - Da representação intermédia à linguagem alvo
  - Geração e otimização de código

Compiladores

- Os passos de análise são comuns a muitas outras ferramentas
  - *Pretty printers, syntax highlighters*
  - Interpretadores
    - Linguagens de programação
    - Linguagens de descrição (html, postscript, latex)
    - Linguagens de consulta a bases de dados
  - Verificadores estáticos
  - Compiladores de silício

Objetivos

Metodologias e técnicas da geração de código para microprocessador a partir de um programa descrito numa linguagem de alto nível

- Compreender os objetivos e a arquitetura de um compilador
- Compreender os princípios da análise lexical e saber implementar analisadores lexicais, quer de raiz, quer usando ferramentas adequadas

Objetivos

- Compreender os princípios da análise sintática e saber implementar analisadores sintáticos descendentes e ascendentes, quer de raiz, quer usando ferramentas adequadas
- Compreender os princípios da análise semântica e saber implementar esse tipo de análise
- Compreender alguns princípios da geração do código final

Programa

- Objetivos e arquitetura de um compilador
- Análise lexical
- Análise sintática (descendente e ascendente)
- Sintaxe abstrata
- Análise semântica
- Registos de ativação
- Representação intermédia
- Geração de código

Competências genéricas

Principais

- Instrumentais
  - Capacidade de análise e síntese
  - Conhecimentos de informática relativos ao âmbito do estudo
- Pessoais
  - Raciocínio crítico
- Sistémicas
  - Preocupação com a qualidade
  - Aplicação prática de conhecimentos teóricos

Competências genéricas

Secundárias

- Instrumentais
  - Comunicação oral e escrita
  - Capacidade de resolução de problemas
- Pessoais
  - Capacidade de trabalho em grupo
- Sistémicas
  - Capacidade de aprendizagem autónoma

Métodos de ensino

Aulas teóricas

- Exposição da matéria pelo docente
- Esclarecimento de dúvidas de interesse geral para a turma

Aulas teórico-práticas

- Consolidação dos conceitos teóricos
- Realização de exercícios teóricos e práticos e apresentação das diferentes etapas do projeto

Métodos de ensino

Aulas de prática laboratorial

- Apoio à realização do projeto e dos exercícios propostos nas aulas teórico-práticas

Trabalho independente

- Revisão das matérias lecionadas
- Realização não acompanhada de exercícios e das tarefas do projeto

Métodos de ensino

Acompanhamento dos estudantes

- Realizado preferencialmente durante as aulas teórico-práticas e de prática laboratorial
- Dúvidas relativas à unidade curricular (conteúdos e/ou funcionamento) durante o horário de atendimento dos docentes

Pré-requisitos

Pressupõem-se adquiridos os conhecimentos básicos de:

- Programação em C
- Algoritmos e Estruturas de Dados
- Linguagens Formais e Autómatos

Unidades curriculares:

- Princípios de Programação Procedimental
- Algoritmos e Estruturas de Dados
- Teoria da Computação

Modelo de avaliação

- Regime de avaliação:
  - **Projeto (8 valores):** Cumprimento de metas intermédias (14/mar, 11/abr, 24/abr e *9/mai*) e prestação na defesa oral (.../mai)
  - **Teste final (12 valores):** prova escrita global, sem consulta (.../jun, .../jul)
- Exame – Épocas de Recurso e Especial
  - Prova escrita, nos mesmos moldes do Teste Final (12 valores)
- Mínimos: 40% tanto na nota final do projeto (após defesa) como na prova escrita (teste final ou exame).

Modelo de avaliação

- Em caso de fraude
  - Reprovação imediata à disciplina de *todos* os envolvidos
  - Comunicação às instâncias superiores nos termos da regulamentação aplicável
- Exemplos de fraude na componente de Projeto
  - Partilha de trabalhos entre estudantes de grupos distintos, incluindo realização do projeto em “grupos de grupos”
  - Publicação de trabalhos em repositório aberto (GitHub ou outro) antes do lançamento das classificações finais em pauta

Bibliografia

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd ed.)*, Addison-Wesley, 2006.
  - Introdução aos compiladores – Secções 1.1, 1.2
  - Análise léxica – Sec. 3.1, 3.3, 3.4, 3.6, 3.7, 3.8
  - Análise sintática – Sec. 4.1-4.6, 4.8
  - Análise semântica – Sec. 5.1-5.3, 6.3, 6.5
  - Geração de código – Sec. 6.2, 7.1-7.4, 8.1-8.3, 8.6
  - Optimização – 8.5, 8.7, 9.1-9.4
  - Aulas práticas – Capítulo 2 e Apêndice A

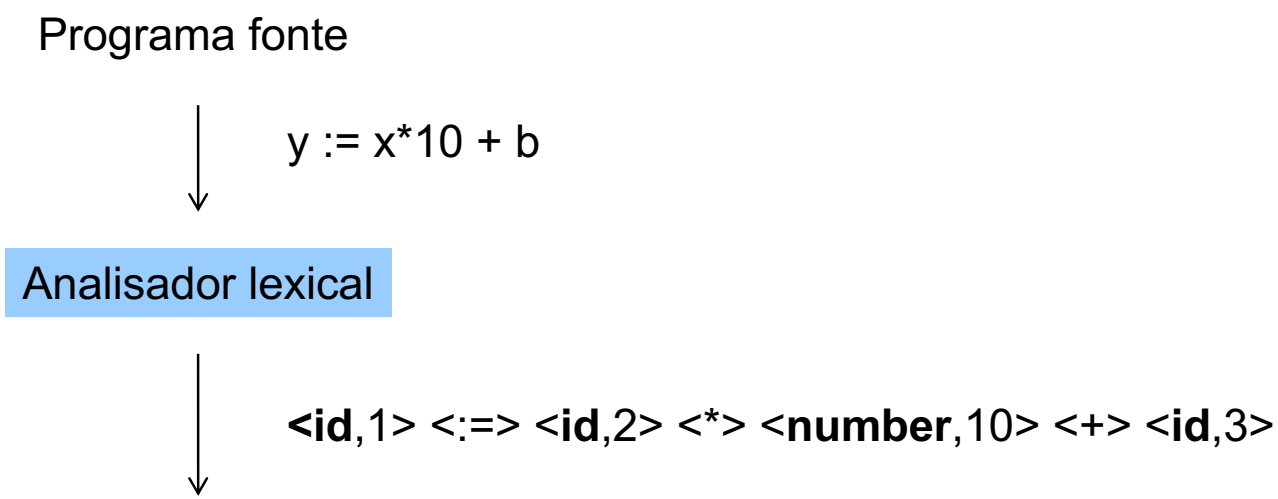
Bibliografia extra

- Keith D. Cooper and Linda Torczon, Engineering a Compiler, 3rd ed., Morgan Kaufmann, 2022.
- Andrew W. Appel. Modern Compiler Implementation in C, Cambridge University Press, 1998.
- Dick Grune, Henri E. Bal, Ceriel J. H. Jacobs, and Koen G. Langendoen, Modern Compiler Design, John Wiley & Sons, Ltd, 2000.
- Rui Gustavo Crespo. Processadores de Linguagens, da concepção à implementação. IST Press. 2001.

Arquitetura de um compilador

- Conceptualmente, um compilador opera em fases
- Cada fase converte o programa de uma representação para outra
- Na prática, nem todas as fases são desenhadas separadamente
- Nesses casos, as correspondentes representações de interface não são construídas explicitamente

Fases de compilação



Fases de compilação

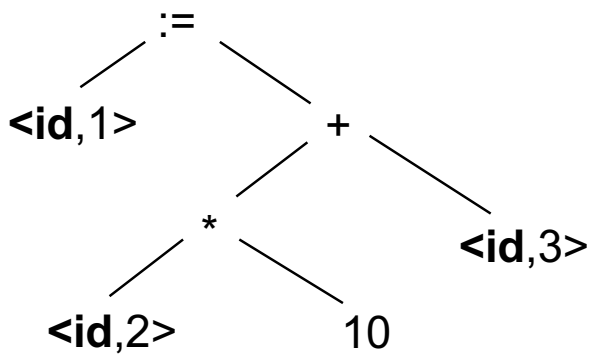
Programa fonte

y := x*10 + b

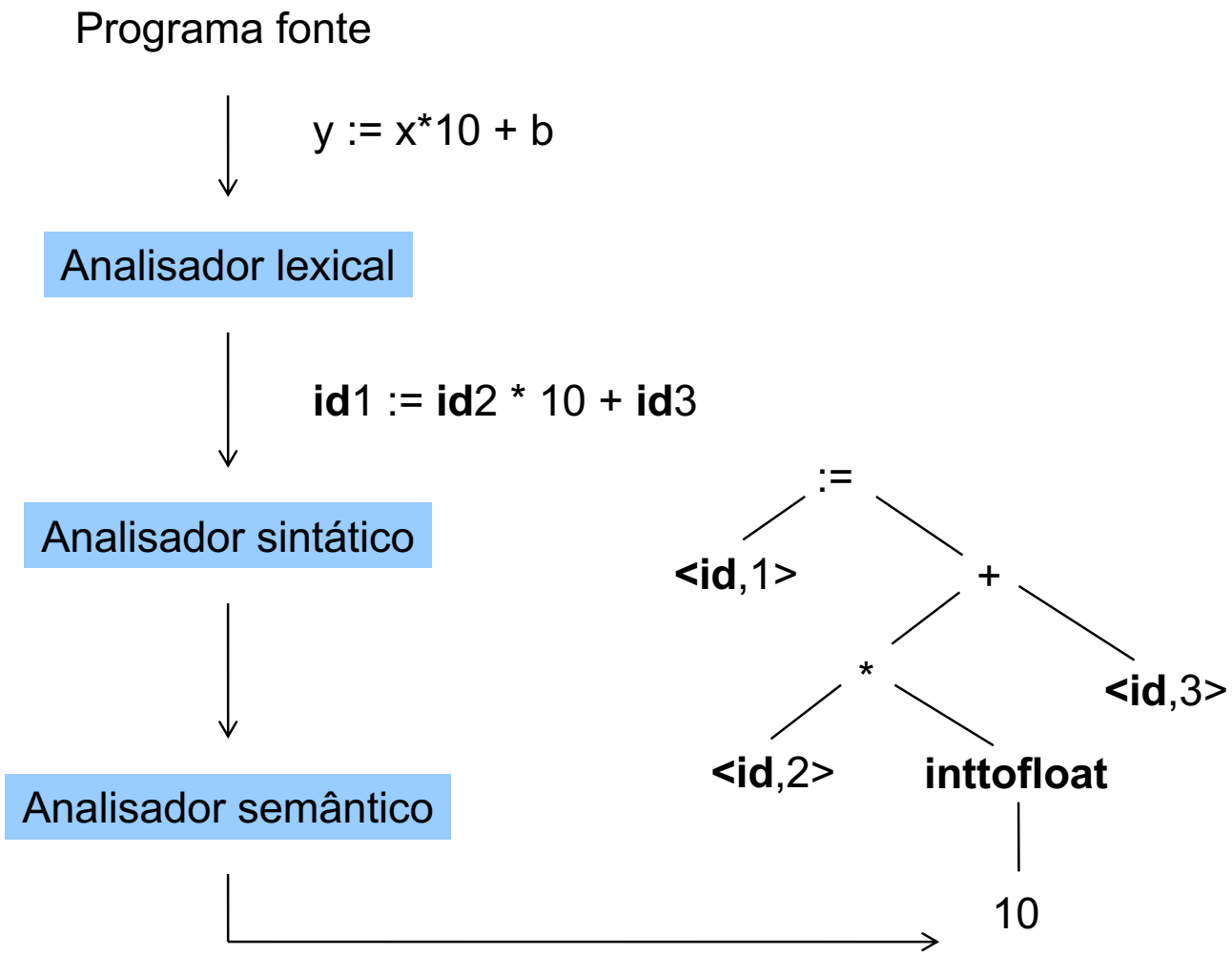
Analisador lexical

<id,1> <:=> <**id**,2> <*> <**number**,10> <+> <**id**,3>

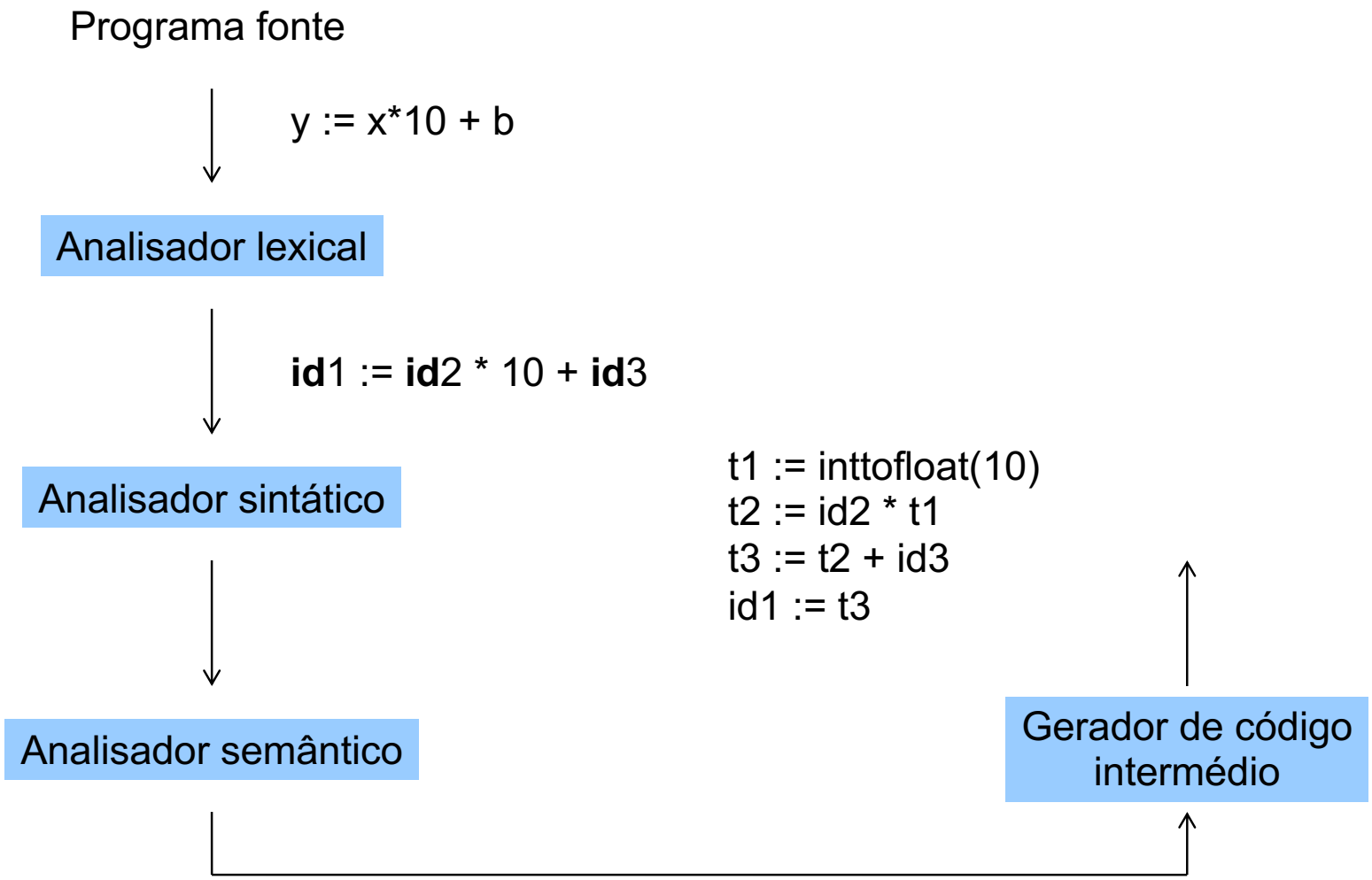
Analisador sintático



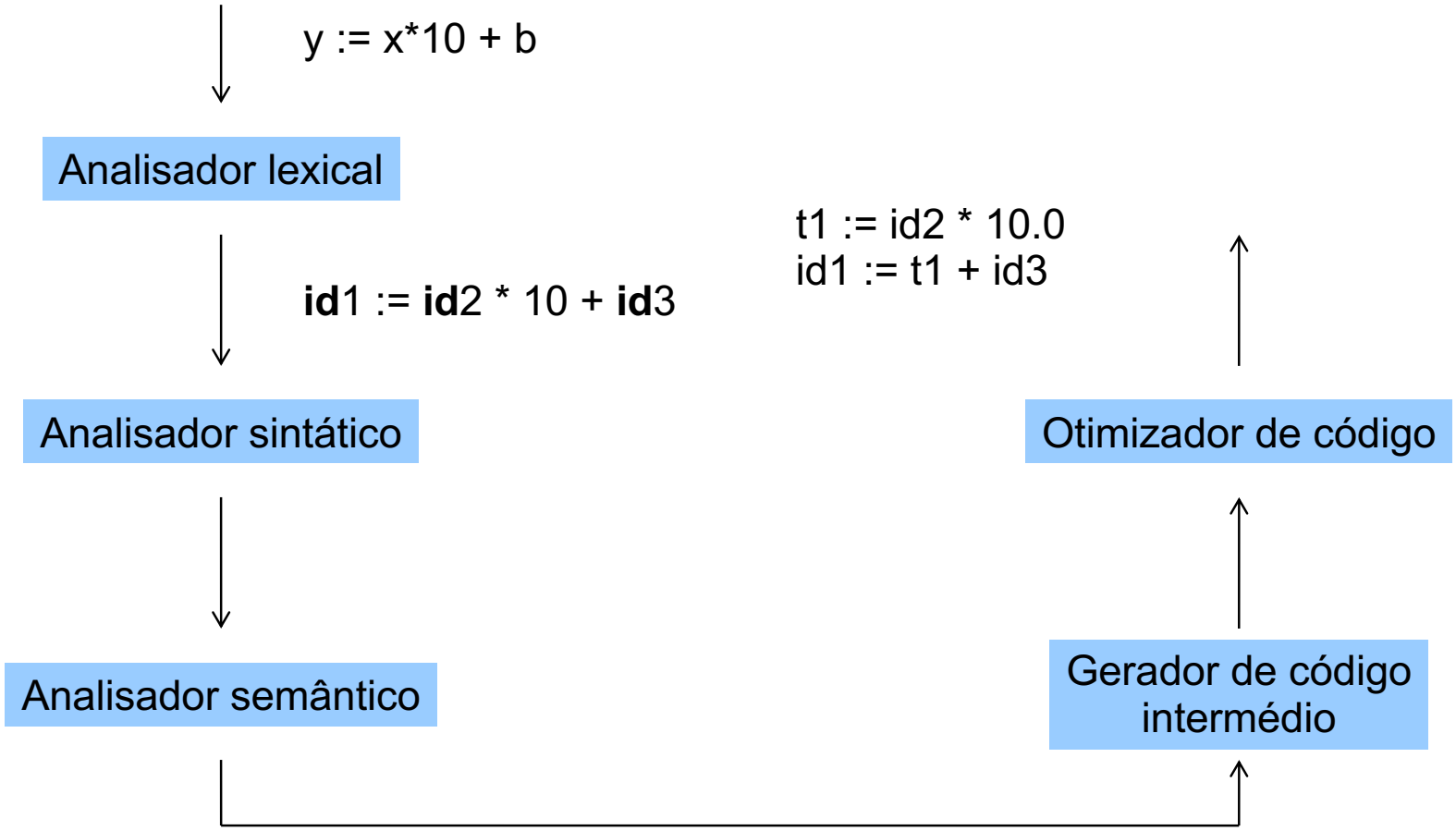
Fases de compilação



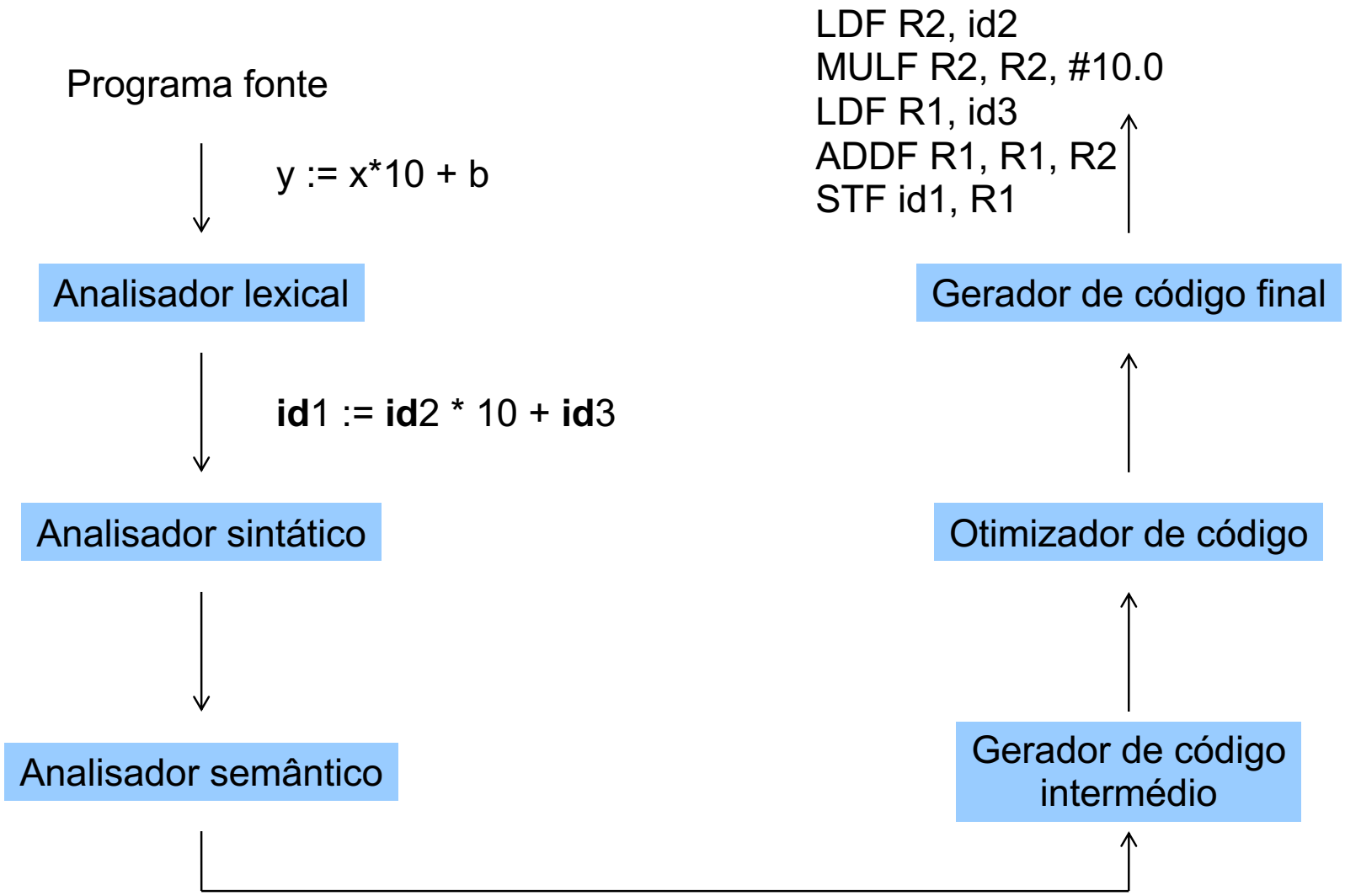
Fases de compilação



Fases de compilação



Fases de compilação



Construção de Compiladores

- Existem diversas ferramentas de apoio à escrita de compiladores
- As mais comuns são:
  - *Scanner generators* – produzem analisadores lexicais a partir de uma especificação baseada em expressões regulares
  - *Parser generators* – produzem analisadores sintáticos a partir de uma especificação baseada numa gramática sem contexto
- Serão usadas na parte prática