Formatos Intermediários

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Junho, 2017

Sumário

Fases

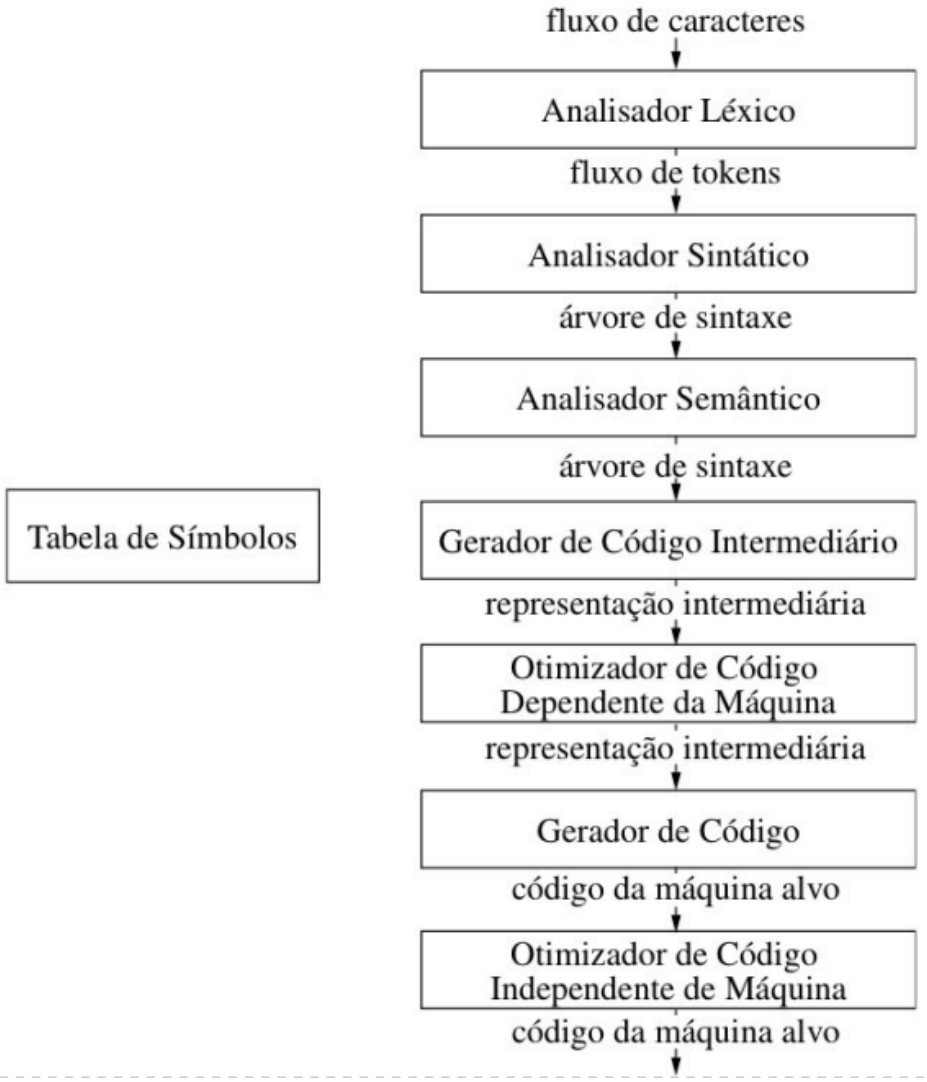


Figura 1.1: Fases do compilador [Aho et al., 2007]

Árvore de sintaxe

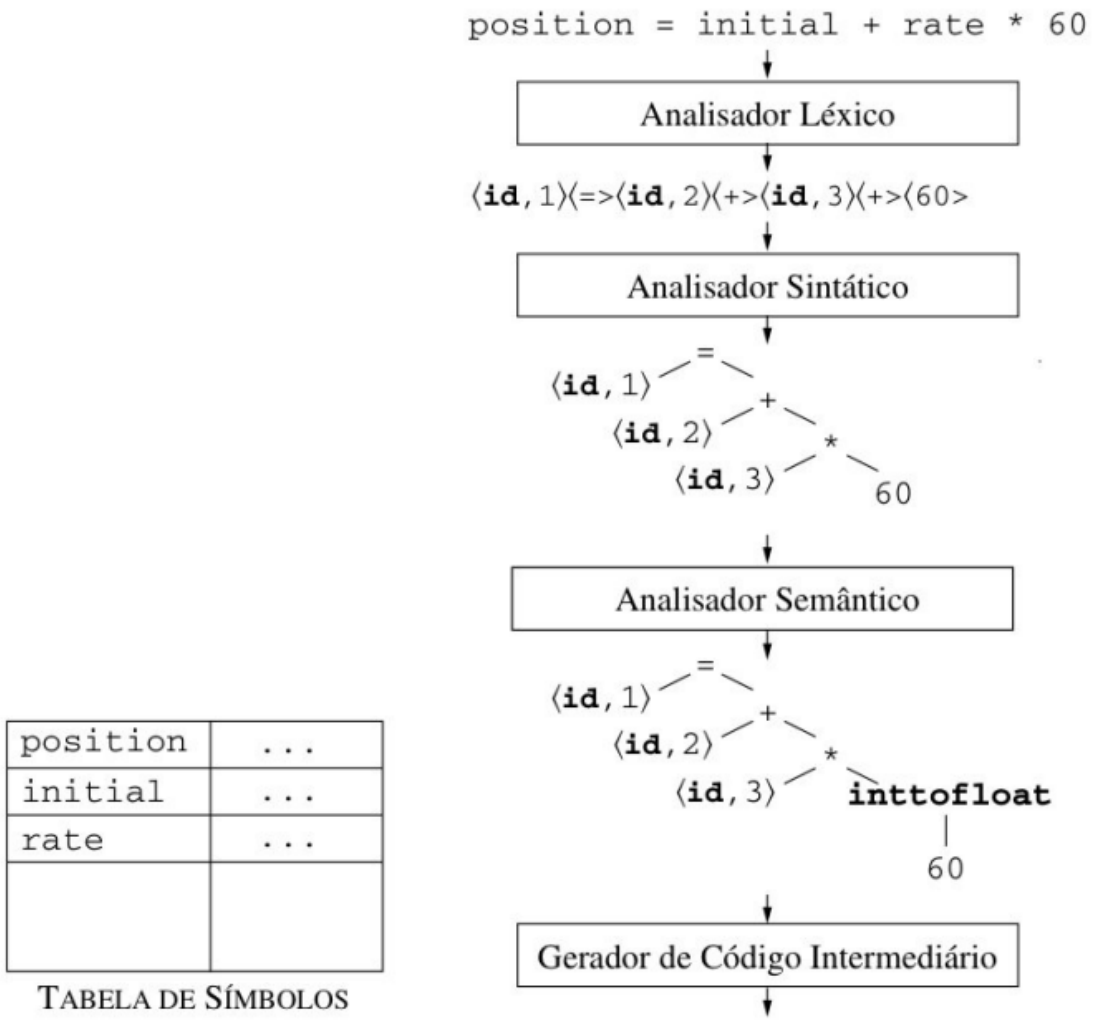


Figura 1.2: Modelo de tradução e atribuição [Aho et al., 2007]

Introdução

- O papel da análise semântica é produzir uma lista de tarefas;
- Para produzir a lista de tarefas o compilador vai precisar da lista de símbolos;
- Nesse momento também se faz a verificação de tipos;
- Objetivo: facilitar a tradução da lista de tarefas em linguagem de máquina.

Etapas da compilação

Análise léxica Detecta entradas com caracteres inválidos;

Análise sintática Produz a árvore de parsing e verifica erros de formação da árvore;

Análise semântica Última fase do "*front end*", detecta os erros que ainda podem existir.

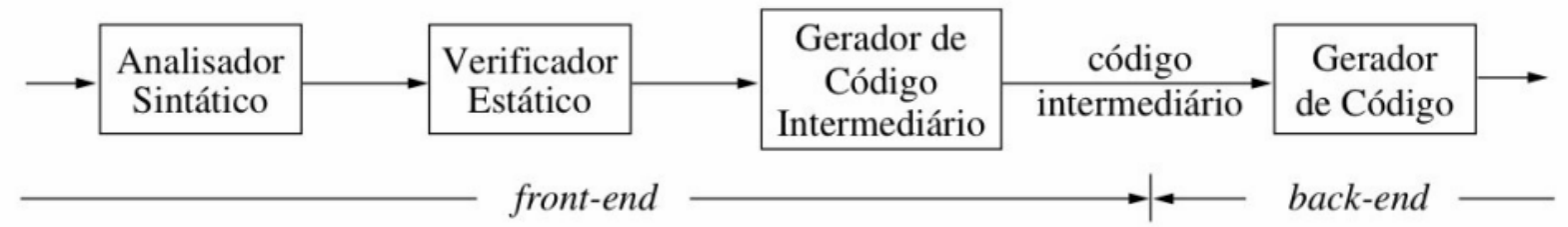


Figura 1.3: Estrutura lógica do front-end do compilador [Aho et al., 2007]

Código Intermediário

- Verificação estática: garante atribuição correta de tipos;
- Produção de uma sequência de representações intermediárias;
- Também realiza uma sequência de passos para compilar o programa.

Figura 1.4: Sequência de representações intermediárias [Aho et al., 2007]

Alto Nível

- A representação intermediária de alto nível representa a primeira passagem pela árvore após a análise sintática [Amarasinghe and Rinard, 2010];
- Preserva a estrutura dos objetos;
- Preserva a estrutura dos fluxos de controle ;
- Objetivo principal: analisar o programa.

Baixo Nível

- A representação intermediária de baixo nível coloca o programa em uma sequência para envio ao processador;
- Move os modelos de dados para o espaço de endereçamento;
- Elimina a estrutura dos fluxos de controle;
- Utilizada em tarefas de compilação de baixo nível:
  - Alocação de registradores;
  - Seleção de instruções.

Tarefas de compilação

1. Determinar formato de objetos e arrays;
2. Determinar a ordem da pilha de execução;
3. Gerar código para ler os valores;
   - `this`
   - `array`
   - `objects`
   - parameters
   - etc
4. Gerar código para avaliar as expressões;
5. Gerar código para escrever valores;
6. Gerar código para as estruturas de controle.

Tabelas de símbolo

- As tabelas de símbolo são utilizadas para produzir [Amarasinghe and Rinard, 2010]:
  - O *layout* dos objetos na memória
  - Código para:
    - Acessar os campos dos objetos;
    - Acessar as variáveis locais;
    - Acessar os parâmetros;
    - Chamar métodos.

Tabela de símbolos no parsing

- As tabelas de símbolo mapeiam identificadores para descritores;
- Operação básica: search
- Dado um símbolo, encontre um descritor;
- Exemplos:
  - Dado um nome de classe, encontre um descritor;
  - Dada uma variável, encontre um descritor;
    - Descritor local;
    - Descritor de parâmetro;
    - Descritor de campo.

Hierarquia em tabela de símbolos

- A hierarquia vem de:

  Escopos aninhados Escopo local dentro do escopo do campo;

  Herança Classe filho dentro da classe pai.

- A hierarquia da tabela de símbolos precisa refletir essa hierarquia;
- Em uma operação de `search` é necessário percorrer toda a hierarquia da árvore para encontrar o descritor.

Descritores

- O que contém um descritor?
- A informação é utilizada para geração de código e análise semântica:

  descritores locais nome, tipo, *stack offset*;

  descritores de campo nome, tipo, *object offset*;

  descritores de método características:
  - Assinatura (tipo de retorno, recebedor, parâmetros);
  - Referência à tabela de símbolos local;
  - Referência ao código do método.

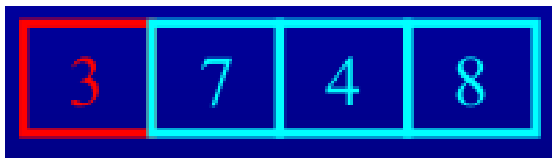
Programa exemplo

Listing 1: Exemplo da classe `Vector` [Amarasinghe and Rinard, 2010]

```
class vector {
    int v[];
    void add(int x) {
        int i;
        i = 0;
        while (i < v.length) { v[i] = v[i]+x; i = i+1; }
    }
}
```

Representando arrays

- Itens armazenados de maneira contígua na memória;
- Tamanho (length) armazenado na primeira palavra.



- Código de cores:
  - Vermelho: gerado automaticamente pelo compilador;
  - Azul, amarelo, violeta: dados do programa ou código;
  - Magenta: executando código ou dados.

Representando objetos

- A primeira palavra aponta para as informações da classe:
  - Tabela de métodos;
  - Dados do Garbage Collector;
  - etc.
- As próximas palavras contém os campos;
- Em vetores, a próxima palavra referencia o campo.

Figura 3.1: Representação de informações da classe [Aho et al., 2007]

Chamando o método

Mostra a execução do método Vector Add.

Hierarquia do Vector Add

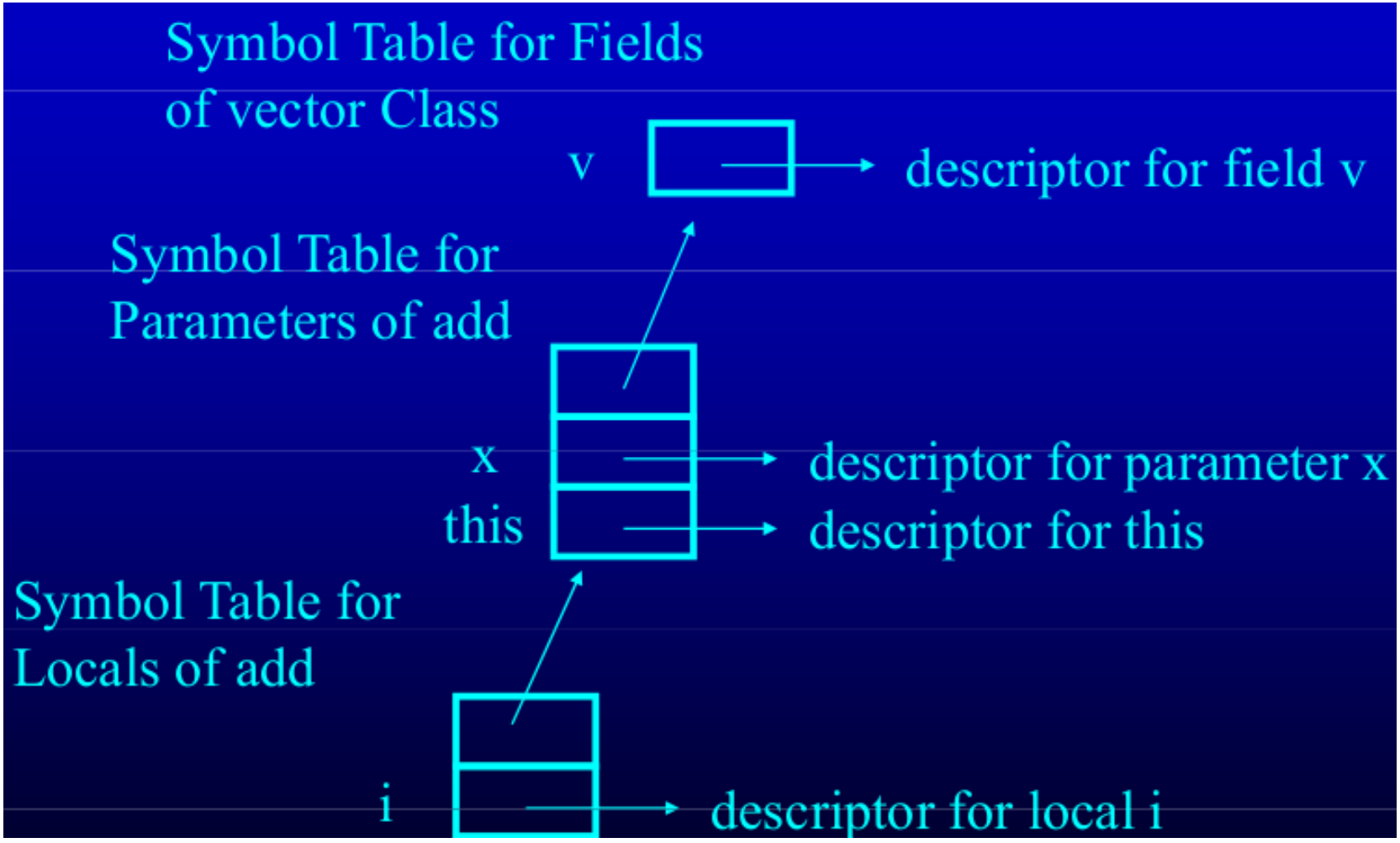


Figura 3.2: Representação da hierarquia no método Vector Add [Aho et al., 2007]

Busca

Exemplo de busca no Vector Add.

OBRIGADO!!!
PERGUNTAS???


Aho, A., Lam, M., Sethi, R., and Ullman, J. (2007).
Compiladores–Princıpios Técnicas e Ferramentas.
Pearson, 2a. edition.


Amarasinghe, S. and Rinard, M. (2010).
Computer language engineering.
Disponível em `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-035-computer-language-engineering-spring-2010/` Acessado em 02/08/2016.