INE5318
Construção de Compiladores

Ricardo Azambuja Silveira
INE-CTC-UFSC
E-Mail: silveira@inf.ufsc.br
URL: www.inf.ufsc.br/~silveira

Identificação da disciplina

- Código: INE 5426
- Nome: Construção de Compiladores
- Horas/aula:
  - teóricas: 20
  - Práticas: 52
  - Total: 72
- pré-requisito: INE 5421 – Linguagens Formais

Plano de Ensino

- OBJETIVO GERAL:
  - Dotar o aluno de conhecimento básico dos conceitos e técnicas necessários para a construção de compiladores, bem como para a compreensão dos conhecimentos envolvidos no projeto de linguagens de programação e o tratamento computacional de linguagens em geral
-

Plano de Ensino

- **OBJETIVOS ESPECÍFICOS:**
  - **Compreender os aspectos ligados ao projeto de linguagens de programação**
  - **Descrever a organização arquitetural dos compiladores e e seu funcionamento**
  - **Compreender e implementar os principais algoritmos de análise léxica.**
  - **Compreender e implementar os principais algoritmos de análise sintática**
  - **Compreender e implementar os processos de análise semântica adotados nos compiladores**
  - **Descrever as técnicas de recuperação de erros utilizadas nos compiladores.**
  - **Identificar as formas de geração e de representação de código intermediário**
  - **Compreender as técnicas de otimização de código e geração de código objeto**
  - **Identificar, avaliar e utilizar ferramentas de apoio na constução de compiladores**

Programa

1) A estrutura de um compilador [1 hora-aula]
2) Linguagens de programação [1 horas-aula]
3) Especificação e projeto de uma linguagem [6 horas-aula
4) Análise léxica [2 horas-aula]
5) Construção de um analisador léxico [8 horas-aula]
6) Análise sintática e correção de erros [6 horas-aula]
7) Construção de um analisador sintático [12 horas-aula]
8) Análise semântica [6 horas-aula]
9) Implementação da análise semântica [12 horas-aula]
10) Geração de código intermediário e otimização [6 horas-aula]
11) Implementação do gerador de código [12 horas-aula]

Metodologia

- Aulas teóricas abordando os conteúdos do programa e aulas práticas orientadas a um projeto de modelagem e implementaçao de uma linguagem e seu correspondente compilador
- Avaliação:
  - 30% da nota através de uma prova escrita
  - 50% da nota relativa as diversas etapas da construçao do compilador apresentadas para a turma ao longo de todo o semestre pelos grupos
  - envolvendo cada uma das etapas que serao avaliadas durante toda a disciplina
  - 20% da nota relativa a participaçao individual nas apresentaçoes de cada etapa da implementaçao do compilador
-

Bibliografia

1. AHO, A.V.; LAM, M. S.; SETHI, R. ULLMAN, J.D. Compiladores – Princípios, Técnicas e Ferramentas, Pearson, 2008
2. DELAMARO, Marcio Eduardo. Como Construir um Complilador Utilizando Ferramentas Java. Novatec, 2004.
3. PRICE, A. M. A., TOSCANI, S. S., Implementação de Linguagens de Programação: Compiladores. Ed Sagra Luzzatto, 1. edição, 2000.
4. WILHELM, Reinhard; MAURER, Dieter. Compiler Design. Ed. Addison-Wesley, 1995, 606p.
5. TREMBLAY, J.P.; SORENSON, P.G. The Theory and Pratice of Compiler Writing. New York: McGraw-Hill, 1985, 796p
6. NETO, João José. Introdução à Compilação. Ed. Livros Técnicos e Científicos, 1987, 222p.

Bibliografia

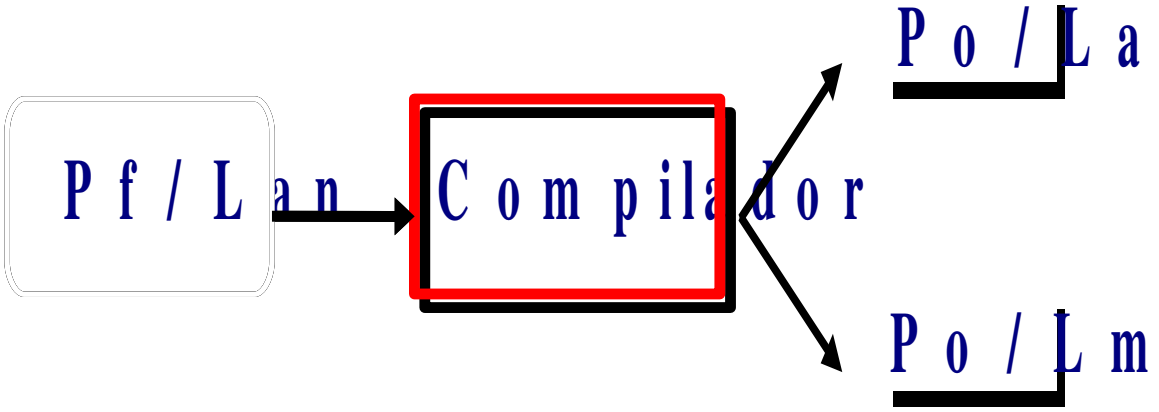
1. WAITE, W. M.; GOOS, G. Compiler Construction. Ed. Springer-Verlag, 1984.
2. KOWALTOWSKI, Tomasz. Implementação de Linguagens de Programação. Ed. Guanabara Dois, 1983, 189p.
3. SETZER, Waldemar; MELO, Inês S. Homem de. A Construção de um Compilador. Ed. Campus, 1983.
4. SEBESTA, R.W. Conceitos de Linguagens de Programação, ed. Bookman, 4. edição, 1999.
5. FURTADO, O. J. V. Apostila de Linguagens Formais e Compiladores - UFSC, 1992.

Definições Preliminares

- Tradutor
  - É um programa que traduz um programa fonte escrito em uma linguagem qualquer (denominada linguagem fonte) para um programa objeto equivalente escrito em outra linguagem (denominada linguagem objeto)

Definições Preliminares

- Compilador
  - É um Tradutor em que a linguagem fonte é uma linguagem de alto nível e a linguagem objeto é uma linguagem de baixo nível (assembly ou máquina)

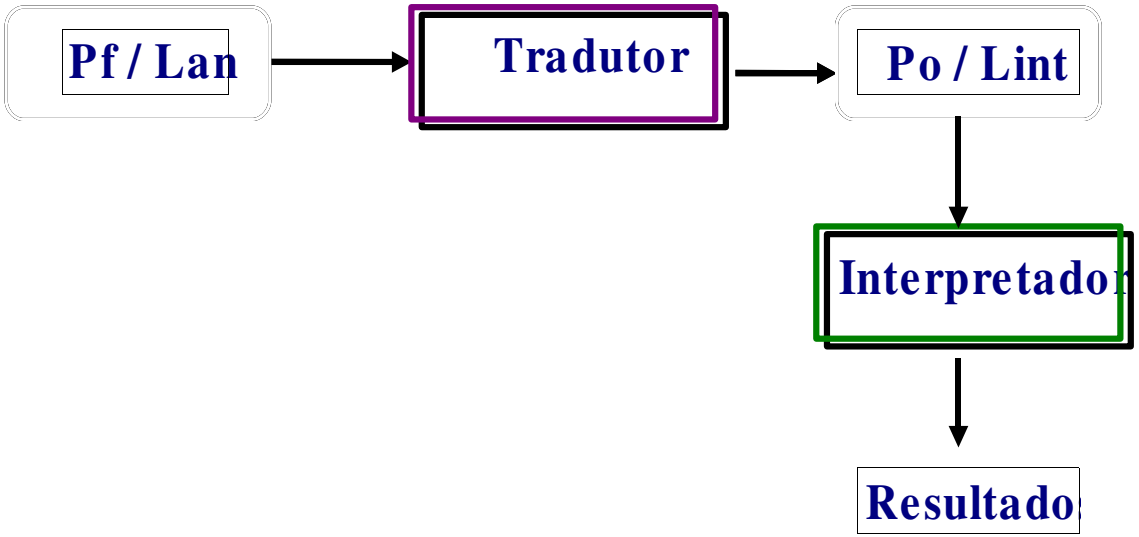


Definições Preliminares

- Interpretador
  - É um programa que interpreta diretamente as instruções do programa fonte, gerando o resultado.

Definições Preliminares

- Tradutor/Interpretador
  - Esquema híbrido para implementação de linguagens de programação



Definições Preliminares

- Montador
  - É um Tradutor em que o programa fonte está escrito em linguagem assembly e o programa objeto resultante está em linguagem de máquina

Definições Preliminares

- Pré-processador
  - É um Tradutor em que tanto o programa fonte quanto o programa objeto estão escritos em linguagens de alto nível

Definições Preliminares

- Cross-Compiler
  - Compilador que gera código para uma máquina diferente da utilizada na compilação.

Formas de Implementação de Compiladores

- Fase
  - Procedimento que realiza uma função bem definida no processo de compilação.
- Passo
  - Passagem completa do programa compilador sobre o programa fonte que está sendo compilado.

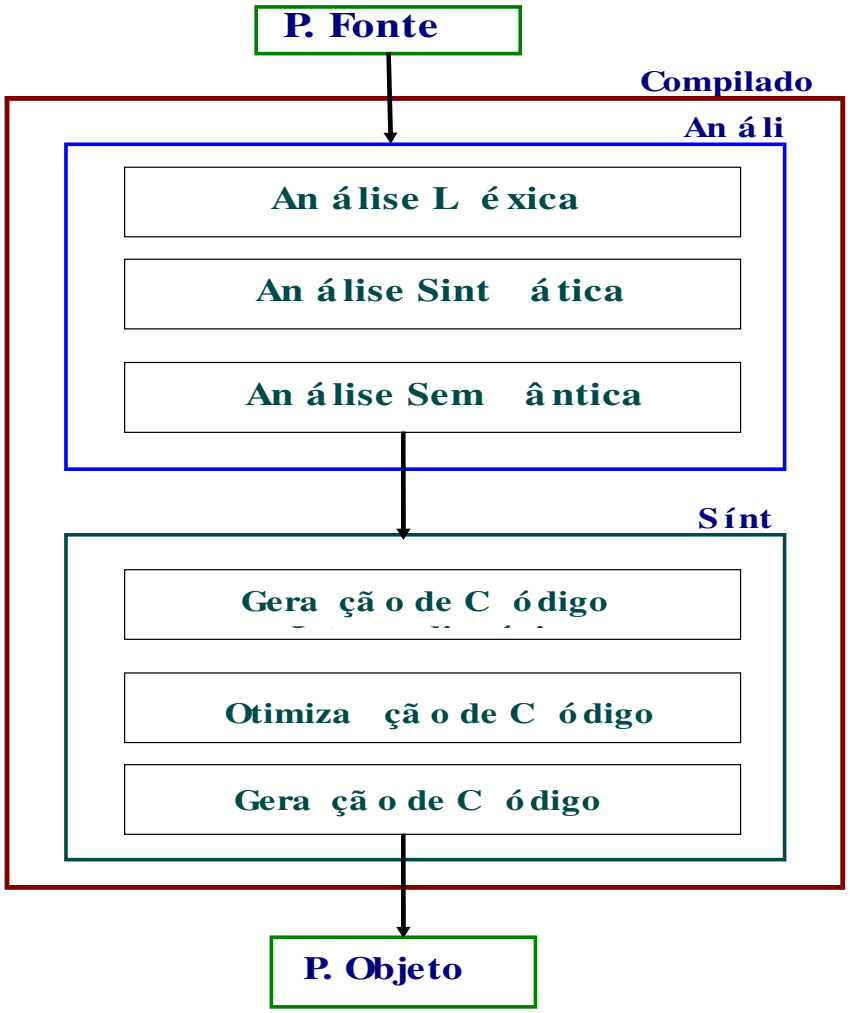
Formas de Implementação

- **Compiladores de 1 e de vários passos**
  - Quantidade de vezes que o P.F. é analisado até que o código objeto seja gerado
  - Diferentes composições
    - (agrupamento de fases)

Critérios de projeto

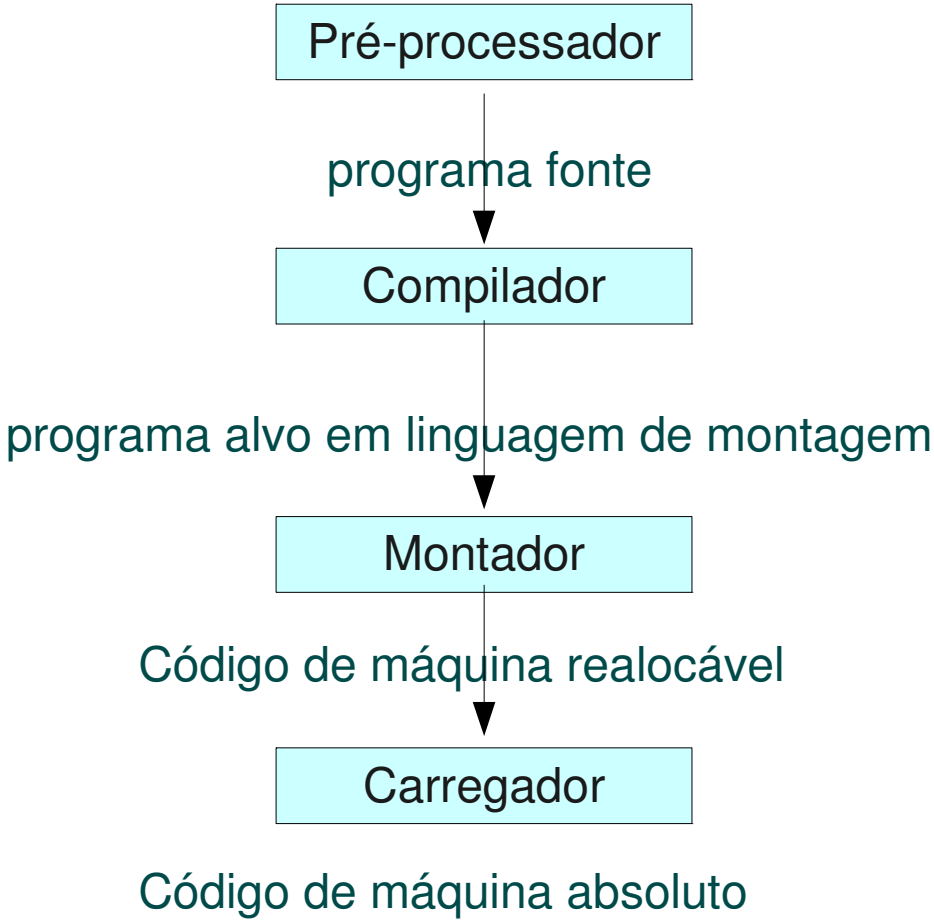
- Memória disponível
- Tempo de Compilação
- Tempo de execução
- Características da Linguagem
- Referências futuras
- Características das Aplicações
- Importância da Otimização
- Tamanho / Experiência da Equipe
- Disponibilidade de Ferramentas de Apoio
- Prazo para desenvolvimento

Estrutura geral de um compilador



O Contexto de um compilador

Esqueleto do programa fonte



Analisador Léxico

- Interface entre o programa fonte e o compilador
- Funções básicas:
  - Ler o programa fonte
  - Agrupar caracteres em itens léxicos (tokens)
    - Identificadores
    - Palavras Reservadas
    - Constantes (numéricas e literais)
    - Símbolos especiais (simples, duplos, ...)
  - Ignorar elementos sem valor sintático
    - Espaços em brancos, comentários e caracteres de controle
  - Detectar e diagnosticar erros léxicos
    - Símbolos inválidos, elementos mal formados
    - Tamanho inválido de constantes, literais e identificadores

Exemplo

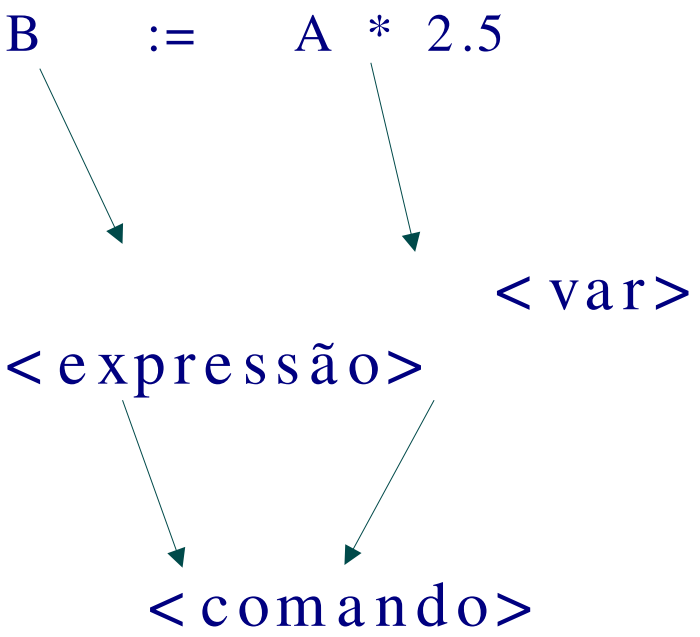
- montante:=deposito_inicial + taxa_de_juros * 60
- Poderiam ser agrupados nos seguintes tokens:

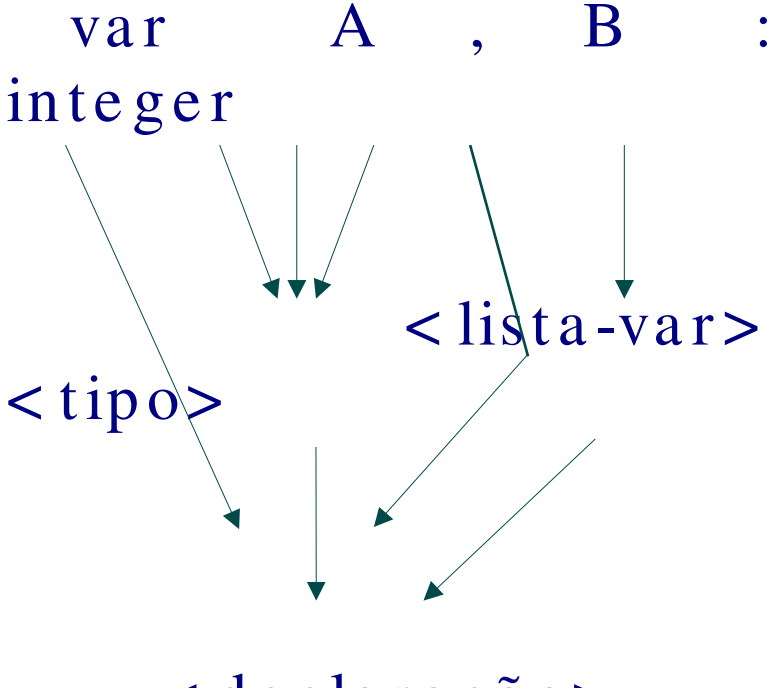
1. identificador **montante**
2. símbolo de atribuição **:=**
3. identificador **deposito_inicial**
4. o sinal de adição **+**
5. o identificador **taxa_de_juros**
6. o sinal de multiplicação *
7. a constante número **60**.

Analisador Sintático

- Funções básicas
  - Agrupar TOKENS em estruturas sintáticas (expressões, comandos, declarações, etc. ...)
  - Verificar se a sintaxe da linguagem na qual o programa foi escrito está sendo respeitada
    - Detectar/Diagnosticar erros sintáticos

Exemplos





Analisador Semântico

- SEMÂNTICA ≅ COERÊNCIA ≅ SIGNIFICADO ≅ SENTIDO LÓGICO
- Funções básicas:
  - Verificar se as construções utilizadas no P.F. estão semanticamente corretas
  - Detectar e diagnosticar erros semânticos
  - Extrair informações do programa fonte que permitam a geração de código

Analisador Semântico

- Verificações Semânticas Usuais em tempo de compilação:
  - Análise de escopo
  - Variáveis não declaradas
  - Múltiplas declarações de uma mesma variável
  - Compatibilidade de tipos
  - Coerência entre declaração e uso de identificadores
  - Correlação entre parâmetros formais e atuais
  - Referências não resolvidas
    - Procedimentos e desvios

Tabela de Símbolos

- Estrutura onde são guardadas as informações (os atributos) essenciais sobre cada identificador utilizado no programa fonte

Atributos mais comuns

- nome
- endereço relativo (nível e deslocamento)
- categoria
  - variável
    - simples - tipo
    - array - dimensões, tipo dos elementos
    - record - campos (quant. e apontadores)
    - ...
  - constante
    - tipo e valor

Atributos mais comuns

- procedimentos
  - procedure ou função
  - número de parâmetros
  - ponteiro para parâmetros
  - se função, tipo do resultado
- parâmetro
  - tipo
  - forma de passagem (valor , referência)
- campo de record
  - tipo, deslocamento dentro do Record

Tratamento e Recuperação de Erros

- Funções
  - Diagnosticar erros léxicos, sintáticos e semânticos encontrados na etapa de análise
  - Tratar os erros encontrados, de forma que a análise possa ser concluída

Gerador de Código Intermediário

- Função
  - Consiste na geração de um conjunto de instruções (equivalentes ao programa fonte de entrada) para uma máquina hipotética (virtual). Exemplo:

Quadrupla	Máquina de acumulador
(+ , A , B , T1) (+ , C , D , T2) (+ , T1 , T2 , E)	carregue A some B armazene T1 carregue C some D armazene T2 carregue T1 multiplique T2 armazene E

Exemplo

montante:=deposito_inicial + taxa_de_juros * 60

temp1 := intoreal(60)

temp2 := id3 * temp1

temp3 := id2 + temp2

id1 := temp3

Otimizador de código

- Função
  - Melhorar o código, de forma que a execução seja mais eficiente quanto ao tempo e/ou espaço ocupado
- Otimizações mais comuns
  - Agrupamento de sub-expressões comuns
    - ex. c := (a + b) * (a + b)
  - Eliminação de desvios para a próxima instrução
  - Retirada de comandos invariantes ao LOOP
  - Eliminação de código inalcançável
  - Redução em força
  - Transformação/avaliação parcial
  - Alocação ótima de registradores

Exemplo

montante:=deposito_inicial + taxa_de_juros * 60

temp1 := intoreal(60)

temp2 := id3 * temp1

temp3 := id2 + temp2

id1 := temp3

Otimizando fica:

temp1 := id3 * 60.0

id1 := id2 + temp1

Gerador de Código

- Função :
  - Converter o programa fonte (diretamente ou a partir de sua representação na forma de código intermediário) para uma sequência de instruções (assembler ou máquina) de uma máquina real.

Exemplo

O trecho:

```
temp1 := id3 * 60.0
id1 := id2 + temp1
```

fica:

```
MOVF   id3, R2
MULF   #60.0, R2
MOVF   id2,  R1
ADDF   R2, R1
MOVF  R1, id1
```