RESUMO COMPILADORES

JOÃO VICTOR BATISTA DE OLIVEIRA

INTRODUÇÃO

O estudo dos compiladores é fundamental para compreender o funcionamento interno das linguagens de programação e o processo de tradução do código-fonte em linguagem de máquina. As Unidades 3 e 4 do livro *Compiladores*, de Regina Fedozzi, abordam as fases finais da análise e da síntese dentro do processo de compilação. A Unidade 3 trata dos mecanismos de **análise sintática, análise semântica, tradução dirigida pela sintaxe e tabela de símbolos**, que garantem a coerência estrutural e lógica do programa. Já a Unidade 4 concentra-se na **geração de código intermediário e código alvo**, bem como nas técnicas de **otimização** e no uso de **expressões regulares**, finalizando o ciclo completo de tradução de uma linguagem de programação.

UNIDADE 3 – TABELA DE SÍMBOLOS, ANÁLISE SEMÂNTICA E TRADUÇÃO DIRIGIDA PELA SINTAXE

3.1 Análise Sintática

A análise sintática é a etapa responsável por verificar se o código-fonte está escrito conforme as regras formais da gramática livre de contexto. Por meio da construção de **árvores sintáticas ou de derivação**, o compilador identifica a estrutura hierárquica das instruções. Nessa fase, também são detectados erros estruturais e ambiguidade, permitindo ao compilador realizar correções e prosseguir com o processo de tradução.

3.2 Tabela de Símbolos

A **tabela de símbolos** é um componente essencial do compilador, atuando como um repositório que armazena informações sobre todos os identificadores do programa — como variáveis, funções, classes e constantes. Cada entrada da tabela contém dados como tipo, escopo, valor e endereço. Todas as etapas do compilador utilizam essa tabela para garantir a **consistência e integridade** das informações durante o processo de compilação.

3.3 Análise Semântica

A análise semântica verifica se as instruções do código fazem sentido dentro das regras da linguagem, avaliando a **coerência e o significado lógico** do programa. Essa etapa assegura que operações matemáticas, atribuições e chamadas de funções sejam semanticamente válidas, identificando erros de tipo, variáveis não declaradas e incompatibilidades. Assim, a análise semântica atua como uma ponte entre a análise sintática e a geração de código.

3.4 Tradução Dirigida pela Sintaxe

A tradução dirigida pela sintaxe é uma técnica que associa **ações semânticas** às produções da gramática. Essas ações são executadas durante a análise sintática e permitem que o compilador gere um **código intermediário** com base na estrutura sintática do programa. Essa abordagem torna o processo de tradução mais eficiente e facilita a integração entre a análise e a geração de código.

UNIDADE 4 – GERAÇÃO E OTIMIZAÇÃO DE CÓDIGO

4.1 Geração de Código Intermediário

Nesta etapa, o compilador converte o programa analisado em uma **representação intermediária**, independente de máquina. Essa representação facilita a **portabilidade** do compilador e permite aplicar técnicas de otimização antes da criação

do código final. Um exemplo prático é o **bytecode** usado na máquina virtual Java (JVM), que pode ser executado em diferentes plataformas sem modificações.

4.2 Geração de Código Alvo

A geração de código alvo é a fase em que o compilador transforma o código intermediário em **instruções de máquina específicas** para o hardware de destino. Esse processo envolve decisões sobre **alocação de registradores**, **ordenação de instruções** e **endereçamento de memória**. O resultado é um arquivo executável pronto para ser interpretado pelo processador.

4.3 Otimização de Código

A otimização busca melhorar o desempenho do código gerado sem alterar seu comportamento. Essa etapa pode ser dividida em:

- **Otimização local**, que atua em pequenos blocos de código;
- **Otimização global**, que abrange todo o programa.
  As principais técnicas incluem **eliminação de redundâncias**, **redução de tempo de execução**, **melhor aproveitamento de registradores** e **minimização do consumo de memória**.

4.4 Expressões Regulares (REGEX)

As expressões regulares são padrões utilizados para **reconhecer cadeias de caracteres** e **identificar tokens** na análise léxica. Também são aplicadas em diversas ferramentas de processamento de texto e verificação de padrões, sendo um recurso essencial na implementação de compiladores modernos.

CONCLUSÃO

As Unidades 3 e 4 completam a compreensão das fases internas de um compilador, abordando tanto os aspectos de **análise estrutural e semântica** quanto as etapas de **síntese e otimização**. A Unidade 3 mostra como o compilador interpreta e valida a estrutura lógica de um programa, enquanto a Unidade 4 evidencia o processo de conversão dessa estrutura em um código eficiente e executável. Juntas, essas unidades permitem ao estudante compreender o funcionamento integral de um compilador e reconhecer sua importância na construção e evolução das linguagens de programação, destacando o equilíbrio entre teoria, prática e eficiência computacional.