UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CONSTRUÇÃO DE COMPILADORES

PROFESSOR DOUTOR LUCIANO FERREIRA DA SILVA

NATÁLIA RIBEIRO DE ALMADA

TRABALHO FINAL - COMPILADOR

BOA VISTA
JULHO DE 2023

NATÁLIA RIBEIRO DE ALMADA

TRABALHO FINAL - COMPILADOR

Projeto de criação de Compilador a ser apresentado ao Centro de Ciência da Computação, da Universidade Federal de Roraima, como requisito parcial para obtenção de nota na disciplina obrigatória do Curso de Ciência da Computação – DCC605 Construção de Compiladores sob orientação do professor Dr. Luciano Ferreira Silva.

BOA VISTA

JULHO DE 2023

RESUMO

Um compilador é um software responsável pela tradução de um código fonte escrito em alguma linguagem de programação para um código executável em linguagem de máquina de plataformas específicas, tal como um sistema operacional. Realiza análises léxicas e semânticas, otimizações e gera o código objeto final. O compilador é necessário para permitir que os programas sejam executados em diferentes sistemas e processadores, melhorando a eficiência e a portabilidade do software.

Por isso, através das aulas ministradas no semestre 2023.1 da disciplina de Construção de Compiladores, foi possível a construção do código especificado nesta documentação.

Palavras-Chave: Compilador, Código, Análise Sintática, Análise Semântica.

ABSTRACT

A compiler is software responsible for translating source code written in some programming language into executable code in machine language for specific platforms, such as an Operational System. Performs lexical and semantic analysis, optimizations and generates the final object code. The compiler is needed to allow programs to run on different systems and processors, improving software efficiency and portability.
Therefore, through the classes taught in the semester 2023.1 of the Compiler Construction discipline, it was possible to build the code specified in this documentation.

Keywords: Compiler, Code, Lexical Analisis, Semantic Analysis.

ESPECIFICAÇÕES DO PROGRAMA

Este código do programa compilador foi desenvolvido na linguagem Python 3.10.2 na IDE Visual Studio Code, com PatternsJSON para aplicação de padrões de estrutura e validação de dados em JSON, garantindo a consistência e integridade dos dados JSON e para facilitar a interoperabilidade entre sistemas.

VARIÁVEIS

Serão aceitas como variáveis quaisquer números e letras, asim como a sublinha "_", desde que o primeiro termo seja sempre uma letra. Exemplos:

- Aceito: b3_21,c_65_.
- Não Aceito: _b23, 2ab_

COMANDOS

Esta Linguagem suporta operações aritméticas entre as variáveis e números, sob duas consições. A primeira delas sendo a obrigatoriedade de a sentença estar na mesma linha. A segunda delas sendo que a operação deve ser atribuída a alguma ouitra variável. Exemplos:

- Aceito: a= b3_21+ c_65_.
- Não Aceito: a12 + 9

GRAMÁTICA

A gramática utilizadapossui capacidade de analisar operações aritméticas, como soma, subtração, multiplicação e divisão, levando em consideração a precedência dos operadores e a presença de parênteses. Para realizar tal análise, é utilizado um analisador de precedência fraca. Isso significa que a gramática define regras específicas para lidar com a ordem de execução das operações, o que garante que as operações sejam realizadas da forma correta, seguindo as convenções e regras da matemática. O uso desse analisador é fundamental para que a expressão aritmética seja interpretada corretamente e produza o resultado esperado.

- Símbolos terminais: +, -, *, /, (,), id, num. Onde 'id'representa um identificador e 'num' representa um número.
- Símbolos não terminais: S, X,Y.
- Símbolo sentencial: S.

  $S \rightarrow S+X$

S → S-X

S → X

X → X*Y

X → X/Y

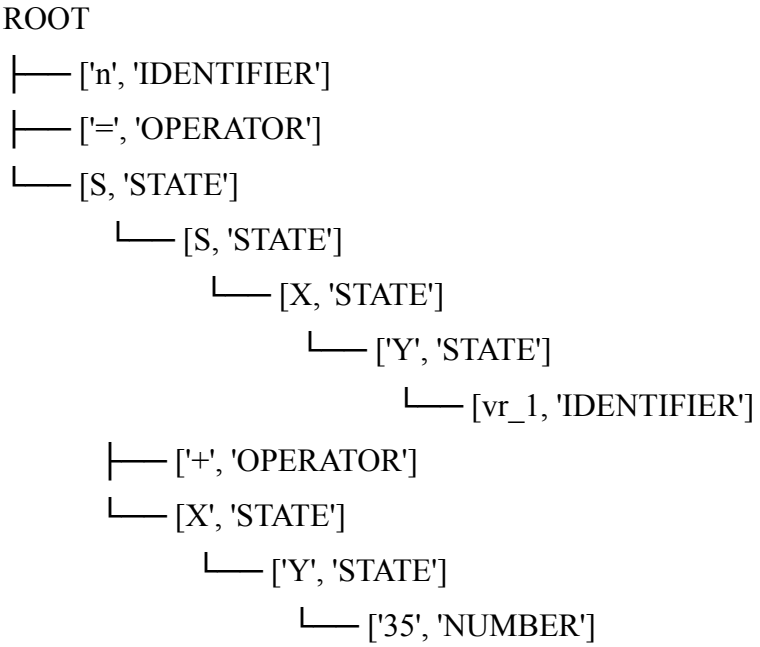
X → Y

Y → (S)

Y → id

Y → num

Exemplo de árvore gerada através de uma linha de código: n = vr_1+ 35



GERAÇÃO DE CÓDIGO INTERMEDIÁRIO

Ocompilador utiliza uma árvore sintática para construir uma tabela de quádruplas. Essas quádruplas são uma forma de representação intermediária de código usada em compilação e otimização de programas. Dessa forma, a árvore sintática é percorrida e, durante o percurso, as quádruplas são construídas e adicionadas à tabela. As quádruplas representam operações e seus operandos, sendo organizadas em quatro campos: operador, operando 1, operando 2 e resultado. Cada operando pode ser uma constante, uma variável temporária ou uma variável normal.

Essas quádruplas representam uma sequência de operações aritméticas que são executadas em ordem. Cada variável temporária _t é usada para armazenar resultados intermediários durante a avaliação da expressão aritmética.

Essa tabela de quádruplas pode ser posteriormente utilizada em etapas de otimização, geração de código final ou outras transformações do programa compilado. Ela fornece uma representação intermediária que simplifica o processo de manipulação e transformação do código.

Exemplo:

[+', '_t1', '32', '_t2']

['*', '12', '_t2', '_t3']

['/', '4', '_t3', 'vr_1']

['-', 'vr_1', '12', '_t5']

['+', '1', '_t5', 'n1']

1. A primeira quádrupla realiza uma adição, utilizando a variável temporária _t1 e o valor 32, e armazena o resultado na variável temporária _t2.
2. A segunda quádrupla realiza uma multiplicação, utilizando os valores 12, _t2 e _t3, e armazena o resultado na variável temporária _t3.
3. A terceira quádrupla realiza uma divisão, utilizando os valores 4, _t3 e vr_1, e armazena o resultado na variável vr_1.
4. A quarta quádrupla realiza uma subtração, utilizando os valores vr_1, 6 e _t5, e armazena o resultado na variável temporária _t5.
5. A quinta quádrupla realiza uma adição, utilizando os valores 1, _t5 e n1, e armazena o resultado na variável n1.

GERAÇÃO EM BAIXO NÍVEL/ ASSEMBLY

O Compilador consegue suportar e converter quatro operações aritméticas em assembly: ADD(+), SUB(-), MUL(*) e DIV(/).

Exemplo:

[+', '_t1', '32', '_t2']	→	ADD _t2, 32, _t1
['*', '12', '_t2', '_t3']	→	MUL _t3, 12, _t2

Instrução de acesso:

1. Entrar no Link do GitHub: https://github.com/nataliaalmada/Compiladores2023.1
2. Ler o README.
3. Seguir as instruções determinadas.